COMMODORE 64

la grafica e il suono

Rita Bonelli
Luciano Pazzucconi
Fabio Racchi
Giovanni Valerio

GRUPPO
EDITORIALE
JACKSON

COMMODORE 64

la grafica e il suono

di
Rita Bonelli
Luciano Pazzucconi
Fabio Racchi
Giovanni Valerio

GRUPPO
EDITORIALE
JACKSON
Via Rosellini, 12
20124 MILANO

INDICE

Gli autori ringraziano James Bachmann, Amministratore Delegato, e Sergio Messa, Direttore Generale, della Commodore Italiana s.p.a., per avere messo a disposizione le apparecchiature e la documentazione necessarie alla realizzazione dell'opera.

Tutti i programmi che compaiono nel testo sono stati provati sul calcolatore e sono memorizzati sulla cassetta allegata al libro. L'editore fornisce a richiesta anche il dischetto contenente tutti i programmi.

I numeri esadecimali che compaiono nel testo, salvo avviso contrario, sono contraddistinti da una H finale. Nelle istruzioni in assembler i numeri esadecimali sono preceduti da $.
Il testo è stato scritto usando il programma di elaborazione testi EASY SCRIPT, su COMMODORE 64. I file di testo sono stati trasmessi tramite RS232, servendosi di software adatto, da COMMODORE 64 a APPLE II, per poter passare i dischetti alla fotocomposizione. Su uno dei prossimi numeri della rivista BIT comparirà un articolo che tratta l'argomento della trasmissione da noi effettuata.

PREFAZIONE

Il presente volume è dedicato alla grafica e al suono. Esso fa seguito al primo volume, dedicato al BASIC, e al secondo dedicato ai file su disco e su cassetta. A volte nel testo compaiono dei riferimenti ai primi due volumi, e, a volte, sono inevitabili alcune ripetizioni.

Il libro è frutto del lavoro di quattro persone; i miei tre collaboratori sono studenti universitari, uno di fisica, uno di ingegneria e uno di scienze dell'informazione. Ognuno di noi possiede un COMMODORE 64 (uno nella versione EXECUTIVE); tutti ne siamo soddisfatti, tenendo naturalmente conto del rapporto costo/prestazioni, e speriamo di aver dato un contributo valido alla conoscenza di questo personal.

Le capacità grafiche e sonore del COMMODORE 64 sono veramente notevoli, come il lettore può rendersi conto anche solo consultando l'indice di questo volume.

Il libro si articola in quattro capitoli. Il primo tratta del video e della tastiera, approfondendone tutti gli aspetti, e mostrando che si può fare della buona grafica anche usando solo i caratteri standard. Il secondo affronta in modo esaustivo l'argomento "grafica" del COMMODORE 64. Il terzo si occupa degli sprite per realizzare grafica e animazione. Il quarto è dedicato al suono.
Seguono tre appendici con le tabelle dei registri del VIC II, del SID e delle frequenze delle note.
Ogni argomento è corredato di programmi esempio commentati, che sono listati nel libro e memorizzati sulla cassetta allegata. Inoltre sulla cassetta sono registrati anche alcuni programmi esempio che nel testo sono solo citati e che costituiscono un ulteriore arricchimento.
Alcuni programmi sono stati realizzati in assembler; ne viene riportato il listato, ma senza entrare naturalmente nel merito del linguaggio.

(Rita Bonelli)

CAPITOLO 1

IL VIDEO E I CARATTERI

1.1 LA TASTIERA

Quando accendi il tuo COMMODORE 64, se tutti i collegamenti sono stati effettuati nel modo corretto, appare sul video il messaggio iniziale seguito dal quadratino lampeggiante, il CURSORE, che ti segnala la disponibilità del calcolatore ad accettare comandi e programmi in BASIC. Nel seguito, salvo avviso contrario, ci riferiamo a un COMMODORE 64 senza alcuna modifica o aggiunta, nè RAM nè ROM.
Per comunicare con il calcolatore usi il più tipico dei dispositivi di ingresso per dati alfanumerici: la tastiera.
Quella del COMMODORE 64 è una tastiera AMERICANA tipo QWERTY (infatti la prima riga di tasti alfabetici comincia con Q, W, E, R, T, Y, mentre nelle tastiere italiane la Z sostituisce la W, dando QZERTY) da 62 tasti, affiancata a una colonna di altri 4 tasti. Nella Figura 1.1 è riportato uno schema della tastiera.

Figura 1.1 Tastiera del COMMODORE 64

La pressione di ciascuno di questi tasti produce un effetto diverso che, se il cursore lampeggiante è presente sullo schermo, viene immediatamente visualizzato. In realtà la pressione di un tasto provoca ben più di un effetto. Questo lo vedremo più avanti. Nella tastiera possiamo distinguere 4 gruppi di tasti:

- alfabetici,
- interpunzione,
- numerici,
- funzione.

TASTI ALFABETICI

Sono 32 e includono, oltre le 26 lettere, anche i simboli:

"@" "*" "↑" "+" "−" "£"

Ad eccezione di" ↑"tutti questi tasti hanno l'aspetto simile a quello mostrato a lato. Nelle condizioni dell'accensione, quando premi il tasto viene generato il simbolo riportato sulla parte alta del tasto ("A"). Il simbolo di destra ("♠") lo ottieni premendo il tasto assieme ad uno dei due SHIFT, posti nell'ultima linea, ai lati della tastiera (corrispondono ai tasti per le maiuscole nelle macchine da scrivere). Il simbolo a sinistra (" r") lo ottieni premendo assieme al tasto quello che reca il simbolo (logo) della COMMODORE, ultimo tasto posto in basso a sinistra sulla tastiera. D'ora in poi ci riferiremo ad esso come "tasto CBM".

Sulla parte frontale del tasto" ↑"compare il simbolo" π ". Lo si ottiene con l'uso di SHIFT

Il tasto "←" (primo in alto a sinistra) si differenzia dagli altri di questo gruppo poichè non genera altri simboli.

TASTI NUMERICI E DI INTERPUNZIONE

Sono 10, tutti sulla prima fila, più altri 5 posti sulla destra della tastiera.

Il loro aspetto è mostrato a fianco. Quando si preme il tasto da solo viene generato il simbolo più in basso tra i due presenti sulla faccia superiore ("1"), mentre quello più in alto ("!") si ottiene con l'uso di SHIFT.

I tasti numerici presentano inoltre, sulla faccia frontale una scritta che per i tasti da "1" a "0" è nell'ordine: BLK, WHT, RED, CYN, PUR, GRN, BLU, YEL, RVS ON, RVS OFF. Tali scritte sono delle

abbreviazioni mnemoniche dei termini inglesi che descrivono l'effetto provocato dall'uso contemporaneo di CTRL e del tasto, che può essere:

- la generazione del codice di colore del cursore, che viene resa immediatamente visibile sullo schermo causando:
  - il cambio immediato del colore del cursore,
  - oppure la comparsa di un carattere di controllo, in campo inverso, corrispondente al colore selezionato.

L'uso contemporaneo del tasto CBM e di uno dei tasti da 1 a 8 genera altri otto codici di colore del cursore. Nella versione EXECUTIVE del COMMODORE 64 sui tasti numerici compaiono anche le indicazioni mnemoniche di questi colori. Lo schema a cui puoi riferirti è riportato nella Tabella 1.1, e potrà esserti utile per decifrare listati di programmi ottenuti con una stampante predisposta per il COMMODORE 64.

TABELLA CORRISPONDENZA COLORI

COLORE	CTRL+	CBM+	CAR.	CHR$
NERO	1		"[illegible]"	144
BIANCO	2		"[illegible]"	5
ROSSO	3		"[illegible]"	28
CIANO	4		"[illegible]"	159
PORPORA	5		"[illegible]"	156
VERDE	6		"[illegible]"	30
BLU	7		"[illegible]"	31
GIALLO	8		"[illegible]"	158
ARANCIONE		1	"[illegible]"	129
MARRONE		2	"[illegible]"	149
ROSSO-CHIARO		3	"[illegible]"	150
GRIGIO-1		4	"[illegible]"	151
GRIGIO-2		5	"[illegible]"	152
VERDE-CHIARO		6	"[illegible]"	153
AZZURRO		7	"[illegible]"	154
GRIGIO-3		8	"[illegible]"	155

Tabella 1.1 Corrispondenza COLORI-TASTI-CARATTERI-CHR$

A questo punto vogliamo farti notare due cose:

1) il carattere REVERSE ON si può ottenere anche con CTRL-R. Questo dipende dal fatto che il SISTEMA OPERATIVO del COMMODORE 64 è stato direttamente derivato da quello dei PET/CBM, calcolatori nati per l'ufficio e la gestione, come quelli della serie 2000, 3000, 4000 e 8000.

2) Il carattere SPAZIO si ottiene premendo la barra spaziatrice. L'uso contemporaneo di uno SHIFT non provoca alcuna differenza visibile, ma il SISTEMA OPERATIVO è perfettamente in grado di notare la differenza tra SPAZIO e SHIFT-SPAZIO.

TASTI DI COMANDO/SERVIZIO
Sono 12 in tutto e passiamo ad esaminarli.

Il tasto CTRL, oltre alla selezione del colore e del modo diretto o inverso del cursore, se premuto durante l'esecuzione di un LISTing o di una stampa da programma, rallenta lo scrolling del video. Se premuto da solo non genera carattere di controllo.

Il tasto RUN/STOP ha tre funzioni fondamentali:

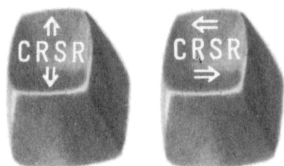
1) premuto da solo permette di interrompere l'esecuzione di un programma BASIC o di un'operazione di sistema, come lettura o scrittura su nastro o altro dispositivo.

2) premuto contemporaneamente a uno SHIFT produce la sequenza di comandi:
LOAD + RETURN + RUN + RETURN nel COMMODORE 64 standard che carica e lancia il primo programma presente su nastro, LOAD "*",8 + RETURN + RUN + RETURN nella versione EXECUTIVE che carica e lancia il primo programma presente sul dischetto.

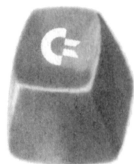
3) premuto contemporaneamente al tasto RESTORE (che genera un segnale riconosciuto dal microprocessore del COMMODORE 64), ripristina (quando possibile) le condizioni iniziali del calcolatore, senza cancellare il programma BASIC, eventualmente presente in memoria.

Il tasto CLR/HOME serve per riportare il cursore nell'angolo in alto a sinistra dello schermo. Premuto insieme a uno SHIFT provoca anche la completa pulizia dello schermo.

Il tasto INST/DEL permette di correggere ciò che si sta scrivendo sullo schermo, CANCELLANDO, se usato da solo, il carattere precedente e spostando indietro di un carattere

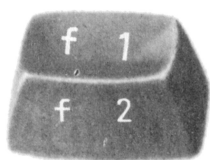
anche il resto della linea sulla quale si trova il cursore; INSERENDO, se usato insieme a uno SHIFT, finchè possibile, uno spazio nella posizione occupata dal cursore e spostando a destra di una posizione il resto della linea logica.

I due tasti CRSR servono per spostare il cursore sullo schermo, orizzontalmente o verticalmente; in giù o a destra, se usati da soli, in su o a sinistra se usati insieme a uno SHIFT.

Sulla tastiera compaiono due tasti SHIFT, uno a sinistra e uno a destra, più un tasto SHIFT/LOCK, che, premuto una volta BLOCCA la funzione degli SHIFT su ATTIVO, premuto una seconda volta ANNULLA la situazione di ATTIVO degli SHIFT.

Il tasto RETURN (da noi abitualmente indicato con CR) quando premuto da solo serve per far accettare al calcolatore una linea di programma, un comando o una serie di dati di INPUT, se premuto insieme a uno SHIFT muove il cursore alla prossima linea logica. Esso non genera mai un carattere di controllo visibile.

Il tasto CBM, come abbiamo già visto, permette di modificare il colore del cursore. Il SISTEMA OPERATIVO è in grado di riconoscere ogni volta che il tasto CBM viene premuto.
Se premi contemporaneamente SHIFT e CBM puoi osservare che sullo schermo avvengono alcuni cambiamenti:

- le lettere MAIUSCOLE diventano MINUSCOLE,
- i simboli GRAFICI di DESTRA dei SOLI tasti alfabetici si trasformano nelle lettere MAIUSCOLE riprodotte sui tasti a cui appartengono,
- gli altri simboli restano invariati.

Ciò succede perchè il tuo COMMODORE 64 dispone di due SET (insiemi o gruppi) di caratteri, normalmente indicati come SET MAIUSCOLO/GRAFICO (o STANDARD CBM) e SET MAIUSCOLO/MINUSCOLO (o COMMERCIALE), che possono essere utilizzati ALTERNATIVAMENTE, cioè a scelta (mai contemporaneamente), a seconda dell'uso che si vuole fare del calcolatore.
Il passaggio da un SET di caratteri all'altro avviene ogni volta che premi SHIFT-CBM.

TASTI FUNZIONE
Sono i 4 tasti disposti sulla destra della tastiera.

In un calcolatore nella sua configurazione base questi tasti non svolgono alcuna funzione particolare, ma poichè essi sono riconosciuti dal SISTEMA OPERATIVO possono essere utilizzati durante la selezione tra diverse possibilità in un programma grazie all'uso, ad esempio, dell'istruzione BASIC GET.
Inoltre, tramite il linguaggio macchina, è possibile assegnare ai diversi tasti funzione un significato particolare.
Su ognuno dei tasti compaiono due scritte: per generare la funzione scritta sopra (funzione dispari) devi premere il tasto da solo, mentre per generare quella scritta frontalmente (funzione pari) devi premere il tasto insieme a uno SHIFT.

Ti suggeriamo di provare a utilizzare tutti i tasti e verificare che il comportamento del calcolatore è quello previsto. Ti accorgerai di diverse cose:

- se premi CR mentre il cursore è su una linea che contiene caratteri messi a casaccio ottieni probabilmente la scritta SYNTAX ERROR e poi READY seguito dal punto; questo non succede se premi SHIFT-CR;
- se cerchi di far scendere il cursore più giù dell'ultima linea dello schermo, i contenuti di questo si spostano verso l'alto, liberando una o due linee in basso, ma perdendo i contenuti delle linee iniziali (effetto chiamato SCROLLING);
- dopo la pressione di alcuni tasti (SHIFT-INST/DEL e SHIFT-2) i tasti di controllo e comando non generano più un effetto immediato sullo schermo, ma fanno comparire dei caratteri in campo inverso.

Perchè succedono queste cose ti sarà chiaro andando avanti nella lettura.

1.2 L'INTERFACCIA CALCOLATORE-UTENTE

1.2.1 Screen Editor

Quando compare il messaggio iniziale di partenza del COMMODORE 64 è attiva quella parte di SISTEMA OPERATIVO che si chiama SCREEN EDITOR e che ti permette di colloquiare semplicemente con il tuo calcolatore, muovendoti come vuoi sullo schermo spostando il cursore per mezzo dei tasti di controllo. L'EDITOR, ogni volta che trovandoti in stato comandi premi il tasto CR, passa il contenuto della LINEA LOGICA sulla quale si trova il cursore all'INTERPRETE BASIC. Quest'ultimo controlla se la linea inizia con un numero, nel quale caso la memorizza come linea di programma nell'area di memoria riservata al programma BASIC; se la linea non inizia con un numero il sistema tenta di eseguirla e, se non ci riesce, segnala gli errori riscontrati. Di tutto questo si tratta diffusamente del volume dedicato al BASIC.
Vediamo ora cosa succede quando tu premi un tasto.

1.2.2 Buffer della tastiera (Keyboard Queue)

Ogni sessantesimo di secondo la tastiera viene scandita per controllare quale o quali tasti sono stati premuti (lavoro svolto dalle routine KERNAL del SISTEMA OPERATIVO) e il codice, generato dal tasto o dalla combinazione dei tasti, viene posto nel byte 197. Se il SISTEMA OPERATIVO non ne ha bisogno immediatamente, tenta di inserirlo, dopo averlo trasformato in ASCII, nella prima locazione libera di una serie di 10 locazioni contigue che inizia al byte 631 e termina al byte 640, detta BUFFER DELLA TASTIERA. Il buffer della tastiera viene gestito come una coda, trattando i caratteri nello stesso ordine nel quale essi vengono inseriti. Per gestire la coda viene usato un PUNTATORE, che risulta il CONTATORE del numero dei caratteri presenti nel buffer, esso è nel byte 198.

1.2.3 Modo tra virgolette (Quote Mode)

Se durante l'introduzione di una linea di comandi o di programma premi SHIFT-2 generi il carattere VIRGOLETTE (APICI) e entri in QUOTE MODE. In questa situazione la pressione dei tasti:
CRSR, CLR/HOME, CTRL-tasto numerico, CTRL-tasto alfabetico (escluso M), CBM-tasto numerico, da F1 a F8, non provoca un effetto immediato, ma la visualizzazione di un carattere in campo inverso: il carattere di controllo corrispondente.
Il QUOTE MODE perdura fino a quando fai una delle cose seguenti:

- premi una volta SHIFT-2, cioè generi un altro carattere VIRGOLETTE: in

effetti il sistema aggiunge 1, ogni volta che generi il carattere virgolette, nella locazione 212; è attivo lo stato QUOTE MODE se il numero di virgolette generato è dispari;

- premi CR; questo chiude la linea e cancella il contenuto della locazione 212;
- premi RUN/STOP-RESTORE: drasticamente questo interrompe la fase di ingresso della linea di comando o programma, pulisce lo schermo, ripristina alcuni puntatori ai loro valori iniziali e ritorna allo stato di EDITOR.

1.2.4 Modo inserimento (Insert Mode)

Se premi SHIFT-INST/DEL mentre stai lavorando in EDITOR, il testo della linea logica a seguito del cursore è spostato a destra di tanti spazi quante sono le pressioni di INST, ammesso che ci sia spazio sulla linea. Questo stesso numero è memorizzato nel byte 216. Puoi allora introdurre altri caratteri, ma devi fare attenzione perchè la pressione dei tasti di comando/servizio e funzione provoca lo stesso effetto di quando sei in QUOTE MODE, con l'eccezione di DEL che produce una T in campo inverso e INST, che, invece, non fa altro che continuare a inserire spazi. Il MODO INSERIMENTO termina con la pressione di CR o con il riempimento di tutti gli spazi creati nel modo stesso.

La Tabella 1.2 che segue mostra l'aspetto dei caratteri di controllo relativi ai tasti di comando/servizio e funzione in QUOTE MODE e in INSERT MODE.

CARATTERI DI CONTROLLO IN QUOTE MODE

TASTO	COME APPARE
CRSR SU	"[illegible]"
CRSR GIU'	"[illegible]"
CRSR SINISTRA	"[illegible]"
CRSR DESTRA	"[illegible]"
CLR/HOME	"[illegible]"
HOME	"[illegible]"
INST/DEL	"[illegible]"
F1	"[illegible]"
F2	"[illegible]"
F3	"[illegible]"
F4	"[illegible]"
F5	"[illegible]"
F6	"[illegible]"
F7	"[illegible]"
F8	"[illegible]"

Tabella 1.2 Caratteri di controllo in QUOTE MODE

Ricordati che la presenza di questi caratteri di controllo in una istruzione di assegnazione (A$ =...) o stampa (PRINT...) produce poi gli stessi effetti che se venissero usati in modo immediato, cioè:

```
PRINT"■I■■O■■■■C■A"
```

produce in stampa: CIAO seguito da READY, così come:

```
PRINT"I■N■G■I■U■,"
```

genera sul video la scritta:

```
I
 N
  G
   I
    U
     ,
```

e:

```
PRINT"■C■O■L■O■R■I"
```

produce la scritta: COLORI, con ogni lettera di un colore diverso, seguita da READY., scritto in verde (con il cursore lampeggiante nello stesso colore).

1.2.5 Il codice ASCII-CBM

Il COMMODORE 64 memorizza i dati di carattere alfanumerico in una forma particolare, utilizzando il codice ASCII-CBM, una delle tante versioni del codice ASCII. Nel Capitolo 7 del volume sul BASIC è riportato l'elenco completo dei codici.
Come puoi notare, nel set maiuscolo/minuscolo al codice 193 corrisponde il carattere A, mentre nel set maiuscolo/grafico al codice 193 corrisponde il carattere grafico a forma di picche. Puoi anche notare che i caratteri corrispondenti ai codici da 96 a 127 sono gli stessi di quelli corrispondenti ai codici da 192 a 223, ma il SISTEMA OPERATIVO li riconosce diversi.

I caratteri stampabili sono 128 più 128 in campo inverso e si ottengono con la pressione dei vari tasti della tastiera in combinazione anche con SHIFT e CBM.

1.2.6 Organizzazione dello schermo e Editor

Per poterti permettere l'interazione che abbiamo visto l'EDITOR deve essere in grado di rilevare cosa è presente sullo schermo in ogni istante e/o modificarlo, perciò deve memorizzare la posizione del cursore sullo schermo, il colore del cursore, il modo (diretto/inverso, quote, inserimento) e altri dati ancora.
Lo schermo è organizzato in una matrice di 25 linee di 40 caratteri ciascuna, rappresentata in memoria da due serie di 1000 locazioni, in ciascuna delle quali le prime 40 locazioni rappresentano la prima linea, le successive 40 la seconda linea e così via.
Le due aree di memoria da 1000 byte ciascuna sono:

1) MAPPA VIDEO: normalmente inizia all'indirizzo 1024, ma questo indirizzo può essere cambiato, come descritto nel seguito (vedi Paragrafo 2.3.2). In essa sono memorizzati i caratteri da visualizzare, utilizzando un codice diverso dal codice ASCII e chiamato DISPLAY CODE (D-CODE).

2) MAPPA COLORE: inizia sempre alla locazione 55296 e termina alla locazione 56295. Non è composta da 1000 byte, ma da 1000 NIBBLE, cioè semibyte, gruppi di 4 bit nei quali sono rappresentabili solo 16 valori diversi.

Le due aree di memoria si possono schematizzare come indicato in Figura 1.2.

Figura 1.2 MAPPA VIDEO e MAPPA COLORI

Puoi accedere a ognuna di queste locazioni di memoria, sia per scrivere che per leggere; l'indirizzo di ognuno dei caratteri sullo schermo lo puoi calcolare sommando all'indirizzo del primo carattere della linea la posizione (0-39) del carattere sulla linea stessa.
Per far comparire un carattere sullo schermo occorre:

1) memorizzare nella MAPPA VIDEO il codice del carattere da visualizzare in D-CODE,

2) memorizzare nella locazione corrispondente della MAPPA COLORE un numero compreso tra 0 e 15, come numero del colore desiderato per il carattere.

Questo è esattamente quello che fa l'EDITOR, con la differenza che anzichè lavorare con le LINEE FISICHE (25 di 40 caratteri ciascuna), esso tratta LINEE LOGICHE, che possono essere lunghe fino a 80 caratteri.
Quando scrivi qualcosa sullo schermo l'EDITOR mantiene traccia dei movimenti del cursore in modo da sapere sempre:

- su quale linea logica dello schermo si trova il cursore, nel byte 214,
- in quale posizione (0-79) sulla linea logica si trova il cursore, nel byte 211,
- quale è l'indirizzo del primo carattere della linea logica del cursore nella mappa video ((byte 210)*256+(byte 209)) e nella mappa colore ((byte 244)*256+(byte 243)).

Il sistema può fare questo mantenendo una TABELLA DEI COLLEGAMENTI TRA LE LINEE (byte 217-242) di 26 byte, ognuno dei quali contiene un numero che indica se tale linea è o non è la prosecuzione della linea precedente. Tale numero è il risultato intero della divisione per 256 dell'indirizzo del primo carattere della linea fisica + 128 se la linea NON E' di continuazione, cioè il byte alto dell'indirizzo del primo carattere sulla linea fisica, eventualmente incrementato di 128. Questa tabella è aggiornata dopo ogni operazione di modifica dei contenuti dello schermo che interessi più di una linea fisica. Ad esempio, quando stai modificando una linea in un listato di programma, più corta di 40 caratteri, nel momento in cui tenti di scrivere il 40-esimo carattere il sistema si accorge che la linea fisica seguente non è parte della linea logica sulla quale stai scrivendo; crea allora sullo schermo una linea bianca, sulla quale tu potrai proseguire a scrivere, mentre la tabella dei collegamenti viene aggiornata.
Per quanto riguarda il calcolo dell'indirizzo del primo carattere di ogni linea il SISTEMA OPERATIVO si avvale di una tabella in ROM che contiene gli indirizzi di inizio delle 25 linee fisiche sullo schermo, quando la mappa video inizia in 1024, e da questa parte per effettuare i calcoli e per trovare tutti gli altri indirizzi e valori necessari.
Al momento in cui premi CR, il contenuto della linea logica sulla quale si trova il cursore (e che le vale la denominazione di LINEA CORRENTE) viene trascritto in una ZONA di memoria chiamata BUFFER DI INPUT, poichè è utilizzata per tutte le operazioni di ingresso dati. Essa inizia alla locazione 512 ed è lunga 89 byte; dato che dalla tastiera vengono letti solo 80 caratteri, in questo caso l'utilizzo non è completo.

L'EDITOR copia INTEGRALMENTE ciò che trova sul video e questo ti permette, ad esempio, tutte le correzioni e modifiche esaminate prima, e in più, di includere nelle stringhe di stampa caratteri di controllo, che normalmente non sono ottenibili da tastiera. Puoi operare così:

- lasciare lo spazio per i caratteri da aggiungere,
- porre il cursore in REVERSE MODE (CTRL-RVS ON),
- posizionarti dove vuoi inserire i caratteri di controllo,
- premere:
  - SHIFT-M, che appare come ▣ e corrisponde a SHIFT-CR,
  - N, che appare come ▣ e corrisponde a SHIFT-CBM nel set maiuscolo/grafico,
  - SHIFT-N, che appare come ▣ e corrisponde a SHIFT-CBM nel set maiuscolo/minuscolo,
  - H, che appare come ▣ e corrisponde alla disattivazione della combinazione SHIFT-CBM,
  - I, che appare come ▣ e corrisponde all'abilitazione della combinazione SHIFT-CBM.

Questi codici sono validi nel set ASCII-CBM e quindi il COMMODORE 64 sa cosa memorizzare.
ATTENZIONE però, dato che SHIFT-CR ha effetto ANCHE NEI LISTATI, esso può rendere difficile, se non impossibile, una correzione della linea che lo contiene.

Quando, muovendoti con il cursore o introducendo una linea, tenti di superare l'ultima linea dello schermo, il SISTEMA OPERATIVO ti procura uno spazio libero, spostando i contenuti dello schermo in su di una linea, se, invece, a eccedere la capacità dello schermo è un LISTATO, allora lo spostamento delle linee, detto scrolling, è di una linea logica, cioè di una o due linee fisiche.

1.2.7 Riferimenti in pagina zero

Mentre svolge le sue attività il SISTEMA OPERATIVO memorizza i dati che gli servono in alcune locazioni riservate, in particolare in quelle locazioni che hanno indirizzo compreso tra 0 e 255 (pagina 0), alle quali il microprocessore del COMMODORE 64 può accedere molto velocemente. Alcune di queste locazioni le abbiamo già viste, ma ce ne sono altre che riguardano la gestione del video e della tastiera.

- 9 — indica in quale posizione dello schermo si deve cominciare a scrivere dopo un TAB;
- 145 — indica la pressione del tasto STOP e di alcuni tasti, come indicato nella Tabella 1.3.

BYTE 145

STOP	Q	CBM	SPAZIO	2	CTRL.	←	1

OGNI BIT CORRISPONDE A UN TASTO
IL BIT E' 0 PER TASTO PREMUTO
IN ASSENZA DI TASTI PREMUTI CONTIENE 255

Tabella 1.3 Contenuto BYTE 145

- 199 <> 0 per RVS ON,
  > 0 per RVS OFF;

- 201-202 al momento di eseguire una INPUT in questi due byte sono memorizzate le coordinate di schermo alle quali fare comparire il cursore;

- 646 colore del cursore; modificare il valore di questo byte equivale ad impostare da tastiera o da programma un codice di controllo del cursore;

- 650 indicazione di ripetizione automatica dei tasti:
  0 per ripetizione solo dei 2 tasti CRSR, del tasto INST-/DEL e dello spazio (valore di default),
  1-127 per disabilitare la ripetizione,
  128-255 per ripetizione abilitata su tutti i tasti;

- 653 tasto ausiliario premuto:
  bit di posizione 0 per SHIFT,
  bit di posizione 1 per CBM,
  bit di posizione 2 per CTRL,
  se i tasti sono premuti contemporaneamente vanno a 1 tutti i bit corrispondenti;

- 657 128 per disabilitare SHIFT-CBM,
  0 per abilitare SHIFT-CBM.

1.3 IL VIDEO

Nel tuo COMMODORE 64 c'è un circuito integrato chiamato VIC II (Video Interface Chip II) che tra le sue funzioni ha quella di attingere le informazioni contenute in certe regioni di memoria e utilizzarle per creare dei segnali che modulati e amplificati raggiungono il televisore o il monitor e creano un'immagine, tipicamente il riquadro per i caratteri circondato dal bordo colorato. Il VIC II è un dispositivo programmabile; questo significa che può funzionare in più di un modo, a seconda dei comandi che gli giungono in ingresso. In particolare, il VIC II può utilizzare le informazioni che legge dalla memoria in molti modi diversi; in questo manuale li vedrai utilizzati tutti.

MODO CARATTERI STANDARD
Quando il VIC II è programmato per utilizzare i dati in memoria per generare un quadro video di 25 linee da 40 caratteri ciascuna, in cui ogni carattere può essere di uno tra i 16 colori possibili, e condivide con gli altri 999 caratteri lo stesso sfondo (anche esso di uno tra i 16 colori possibili), contornato da un bordo colorato (esso pure di uno tra i 16 colori possibili), cioè si trova nella situazione disponibile al momento dell'accensione, diciamo che il COMMODORE 64 si trova nel MODO CARATTERI STANDARD.

1.3.1 Display Code

In modo caratteri, ognuno dei 1000 caratteri visualizzabili sullo schermo può essere scelto tra i 128 stampabili in campo diretto e i 128 stampabili in campo inverso, cioè tra 256 caratteri diversi, ognuno dei quali può essere rappresentato da un codice in un byte della mappa video.
Il codice usato per individuare ognuno dei 256 caratteri è ovviamente diverso dal codice ASCII-CBM, in quanto non comprende i caratteri di controllo, ed è chiamato DISPLAY CODE.
I DISPLAY CODE dei caratteri in campo diretto e inverso si trovano nelle due ultime colonne delle tabelline riportate sul volume del BASIC a partire da pag. 229.
Naturalmente esiste una certa corrispondenza tra CODICE ASCII e DISPLAY CODE; le regole che permettono il passaggio da un codice all'altro sono riportate nella Tabella 1.4.

ASCII	D/CODE
32 <= A <= 63	D = A
64 <= A <= 95	D = A - 64
96 <= A <= 127	D = A - 32
160 <= A <= 191	D = A - 64
192 <= A <= 254	D = A - 128
A = 255	D = A - 161

Tabella 1.4 Relazione tra CODICE ASCII e D-CODE

Per leggere o scrivere un carattere sullo schermo bisogna calcolare l'indirizzo della MAPPA VIDEO in cui eseguire una PEEK o una POKE. Se consideriamo le linee numerate da 0 a 24, dall'alto in basso, e le posizioni dei caratteri da 0 a 39, da sinistra a destra, l'indirizzo si calcola come:

ind.MV=40*num.linea+posiz.car.+ind.base MV
All'accensione l'indirizzo base della mappa video è 1024.

1.3.2 Colore caratteri, sfondo e bordo

In modo caratteri il colore di ognuno dei caratteri può essere scelto tra i 16 disponibili senza alcuna limitazione. Nella mappa colore sono disponibili 4 bit per il colore di ogni carattere e questi bastano per rappresentare un numero tra 0 e 15. La corrispondenza tra codici colore e colori è la seguente:

0..nero
1..bianco
2..rosso
3..ciano
4..porpora
5..verde
6..blu
7..giallo
8..arancione
9..marrone
10...rosso-chiaro
11...grigio-1
12...grigio-2
13...verde-chiaro
14...azzurro
15...grigio-3

si veda anche a pag. 108 del volume sul BASIC.
L'indirizzo in cui eseguire la lettura (PEEK) o la scrittura (POKE) del codice colore si ricava in modo analogo a quello per la mappa video:

ind.MC=55296+40*num.linea+pos.carattere

Quando effettui una lettura devi ricordarti di considerare solo i 4 bit meno significativi del valore ottenuto, ovvero, se A è l'indirizzo della mappa colore di un certo carattere (poichè i 4 bit più significativi, che corrispondono ad altrettante linee non connesse, possono essere valori casuali):

codice colore= PEEK(A) AND 15

Lo stesso sistema di codifica dei colori è usato per lo sfondo comune a tutti i caratteri (detto BACKGROUND COLOR), e per il colore del bordo (detto BORDER COLOR). Per modificare questi due colori bisogna accedere a due diverse locazioni di memoria:

per lo sfondo: per scrivere: POKE 53281, colore
per leggere: C=PEEK(53281) AND 15

per il bordo: per scrivere: POKE 53280, colore
per leggere: C=PEEK(53280) AND 15

L'operazione AND è necessaria poichè anche i primi 4 bit delle due locazioni lette corrispondono a 4 linee non connesse, e vanno quindi mascherati.

1.3.3 Visualizzazione

Il D-CODE di ogni carattere viene utilizzato dai circuiti di controllo del VIC II per ritrovare in memoria la descrizione del carattere stesso per punti. Ognuno dei 512 caratteri visualizzabili sullo schermo (256 per ognuno dei due set disponibili) è descritto in memoria da una sequenza di 8 byte. Nel modo caratteri standard ogni bit 1 corrisponde a un punto nel colore scelto per il carattere, ogni bit 0 corrisponde a un punto nel colore scelto per lo sfondo.
Il D-CODE viene usato come indice per calcolare l'indirizzo di inizio della sequenza di 8 byte (cioè l'indirizzo del primo) nel seguente modo:

ind.=ind.base descr caratt.+8*D-CODE

L'indirizzo base della descrizione dei caratteri, che chiamiamo BC da Base Character, può essere modificato a seconda della necessità dell'utente. Nel modo caratteri standard:

ind.=53248+8*D-CODE nel set maiuscolo/grafico
ind.=55296+8*D-CODE nel set maiuscolo/minuscolo.

Ad esempio, il carattere "&" viene rappresentato nei byte da 53552 a 55559 nel set maiuscolo/grafico, e nei byte da 55600 a 55607 nel set maiuscolo/minuscolo, come segue (il primo numero è la rappresentazione binaria, il secondo quella decimale), e appare come indicato a lato:

```
00000000 ....................... 0 . . . . . . . .
00111100 ..................... 60 . . * * * * . .
01100110 .................... 102 . * * . . * * .
00111100 ..................... 60 . . * * * * . .
00111000 ..................... 56 . . * * * . . .
01100111 .................... 103 . * * . . * * *
01100110 .................... 102 . * * . . * * .
00111111 ..................... 63 . . * * * * * *
```

e il carattere "@" nei byte da 53248 a 53255 e nei byte da 55296 a 55303, così:

```
00000000 ....................... 0 . . . . . . . .
00111100 ..................... 60 . . * * * * . .
01000110 ..................... 70 . * * . . * * .
01001110 ..................... 78 . * * . * * * .
01001110 ..................... 78 . * * . * * * .
01000000 ..................... 64 . * * . . . . .
01000010 ..................... 66 . * * . . * * .
00111100 ..................... 60 . . * * * * . .
```

Se hai letto con attenzione a questo punto penserai: MA QUI C'E' UN ERRORE! Infatti la descrizione del carattere "@" comincia nel set maiuscolo/minuscolo esattamente nella stessa locazione dove inizia la MAPPA COLORE. Come può essere? La spiegazione è che nel COMMODORE 64, a partire dall'indirizzo 53248, coesistono un BANCO di 4K RAM, uno di 4K I/O ed uno di 4K ROM, contenente le descrizioni dei caratteri. Normalmente l'accesso è permesso ai 4K I/O, utilizzati per le operazioni di ingresso e uscita, e le descrizioni dei caratteri in ROM sono inacessibili (prova ad effettuare qualche PEEK di controllo). Quando il VIC II ha bisogno delle descrizioni dei caratteri per generare l'immagine sul video, effettua una operazione detta di "bank switching" (scambio di banchi) che in pratica SOSTITUISCE logicamente il banco di memoria ROM al banco di memoria RAM per il tempo necessario.
Esiste comunque un modo per effettuare lo "switching" da BASIC. Per farlo devi procedere così:

- disabilitare le interruzioni, cioè le chiamate di servizio da parte di altri dispositivi esterni; questo praticamente disabilita la tastiera;
- disabilitare l'ingresso e l'uscita e sostituire il banco RAM con il banco ROM; così risultano bloccate tutte le operazioni di ingresso e uscita.

Per ottenere questo in BASIC, devi scrivere:

1) POKE 56334,PEEK(56334) AND 254

2) POKE 1, PEEK(1) AND 251

a questo punto risulta accessibile la ROM agli indirizzi 53248-57343 e puoi leggere le descrizioni dei caratteri; per riportare il sistema nelle condizioni normali devi eseguire:

3) POKE 1,PEEK(1) OR 4

4) POKE 56334,PEEK(56334) OR 1.

Il cambio del set di caratteri è esemplificato nel programma ES0.

```
 1 REM ES0
 3 REM MOSTRA CARATTERI
 5 BO=53280:SF=BO+1:CO=646
 10 DIMA%(7):POKEBO,14:POKESF,6:POKECO,15
 15 OPEN7,3
 20 INPUT"█RISULTATO SU STAMPANTE (S/N)";R$
 25 AS=1+(R$="N")+2*(R$="S"):IFAS=1THEN20
 30 INPUT"█SET (C)OMMERCIALE O (G)RAFICO";SET$
 33 IFSET$<>"C"ANDSET$<>"G"THEN30
 35 BC=53248-2048*(SET$="C")
 40 INPUT"█D - CODE (0 - 255)";D
 45 IFD<0ORD>255THEN40
 50 PRINT"█";:POKE1024,D:GET#7;A$:A=ASC(A$)
 55 POKE212,0:PRINT"████";
 60 POKE56334,PEEK(56334)AND254
 65 POKE1,PEEK(1)AND251
 70 B=BC+8*D:FORK=0TO7:A%(K)=PEEK(B+K):NEXT
 80 POKE1,PEEK(1)OR4:POKE56334,PEEK(56334)OR1
 90 GOSUB300
 100 IFAS=-1THEN150
 110 INPUT"█ANCORA (S/N)";R$:IFR$="S"THEN20
 120 IFR$<>"N"THEN110
 125 CLOSE7
 130 PRINT"█":END
 150 OPEN4,4,0:CMD4:GOSUB300:PRINT#4:CLOSE4
 155 GOTO110
300 RVS=-(D>127)
305 PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒ D CODE =";
```

```
306 PRINTRIGHT$("   "+STR$(D),4)
310 FORK=0TO7:PRINT"▒";
315 FORJ=0TO7
316 PRINTCHR$(32-81*((2↑(7-J) AND A%(K))<>0));
317 NEXT J
320 PRINT"▒";:IFK<>2THEN330
325 PRINT" ASCII="RIGHT$("   "+STR$(A),4);
326 PRINT" (RVS "MID$("OFFON",1+3*RVS,3-RVS)")";
330 PRINT:NEXT
335 PRINT"▒▒▒▒▒▒▒▒▒▒"
340 RETURN
```

1.4 LAVORARE IN MODO CARATTERI

Anche senza avere a disposizione della grafica sofisticata è possibile scrivere dei programmi utili e interessanti, dai programmi gestionali ai programmi di trattamento testi, a programmi matematici o educativi.
Tutte le volte che è sufficiente un RESPONSO di natura alfanumerica da parte del programma può essere utilizzato il modo caratteri. Riguardo a ciò ci sono almeno tre argomenti da trattare:

1) input controllato,
2) maschere video,
3) presentazione dei dati.

Di questi argomenti abbiamo già parlato nel volume sul BASIC, ma ci sembra utile ritornare a trattarne anche qui, per completezza.

1.4.1 Input controllato

Con questo termine si intende l'introduzione controllata dei dati e questa può essere ottenuta:

A) controllando la validità del dato in ingresso dopo averlo letto;
B) controllando la validità del dato in ingresso durante la sua introduzione.

Il controllo A) può essere fatto dopo una istruzione INPUT, una READ o una INPUT #; normalmente consiste nel controllare che un dato ricevuto soddisfi a determinate condizioni (da notare che, se alcune condizioni non sono verificate, l'errore viene segnalato dal sistema):

- non sia più corto o più lungo di una lunghezzza fissata,
- non contenga caratteri indesiderati, come caratteri di controllo,
- sia del tipo desiderato, cioè contenga solo caratteri alfabetici o solo caratteri numerici (dopo averlo letto come stringa),
- sia limitato in valore, cioè un numero compreso in un intervallo, o un dato alfabetico che rispetti un certo ordinamento,

● soddisfi altri vincoli inerenti all'argomento trattato.

In caso contrario si può agire in diversi modi:

● si segnala l'errore e si richiede una nuova lettura,
● si segnala l'errore e si conclude la procedura in atto,
● si corregge il dato in ingresso e se ne chiede conferma.

Vediamo come è possibile realizzare i diversi controlli.

LUNGHEZZA

Il controllo è effettuato con la funzione LEN(A$), che fornisce la lunghezza della stringa data, come mostrato nel programma ES1.

```
1 REM ES1
110 INPUT"▓▓▓NOME:  ";A$
115 IFLEN(A$)=0ORLEN(A$)>20THEN110
```

Considerando le caratteristiche dell'EDITOR di schermo è conveniente talvolta cancellare prima il vecchio dato, come indicato nel programma ES2.

```
1 REM ES2
5 S$="":FORI=1TO80:S$=S$+" ":NEXTI
110 PRINT"▓▓▓"S$"▓▓";
113 INPUT"NOME:  ";A$
115 IFLEN(A$)=0ORLEN(A$)>20THEN110
```

Il controllo sulla lunghezza del dato può essere eliminato se desiderato, ad esempio quando si include nella stringa di INPUT una serie di caratteri che segnalano la lunghezza desiderata, come mostrato nel programma ES3.

```
1 REM ES3
110 PRINT"▓▓▓";
113 INPUT"NOME:  ------█████████▌";A$
```

PRESENZA CARATTERI INDESIDERATI

Possono essere controllati singolarmente i caratteri della stringa ricevuta, eliminando quelli indesiderati o segnalando errore, come in questo caso, dove il dato in ingresso è in A$, e in uscita si pone E=0 se la stringa è corretta ed E=1 se non lo è, come mostrato nel programma ES4.

```
1 REM ES4
5 REM PROVA ROUTINE CONTROLLO
10 PRINT"♥STRINGA DA CONTROLLARE"
20 INPUTA$:GOSUB100
30 PRINT"STRINGA "MID$("NON CORRETTA",5-4*E)
40 INPUT"QQPREMI [RETURN]";I$
50 RUN
100 REM ROUTINE CONTROLLO
105 FORK=1TOLEN(A$):A1$=MID$(A$,K,1)
110 E=-(A1$<" "ORA1$>"]"ANDA1$<"♠"ORA1$>"♦")
115 IFETHENRETURN
120 NEXT:RETURN
```

Alla linea 110 viene assegnato ad E un valore che è 1 se l'espressione logica tra parentesi (che effettua il controllo sulla non validità del carattere puntato dall'indice K in A$) è vera, 0 se essa è falsa.
Se E=1 il ciclo viene concluso. E' possibile segnalare la posizione del carattere errato memorizzandola in una variabile P, come indicato nel programma ES4.1.

```
1 REM ES4.1
5 REM PROVA ROUTINE CONTROLLO
10 PRINT"♥STRINGA DA CONTROLLARE"
20 INPUTA$:GOSUB100
25 IFETHENPRINTSPC(1+P)"↑"
30 PRINT"STRINGA "MID$("NON CORRETTA",5-4*E)
40 INPUT"QQPREMI [RETURN]";I$
50 RUN
100 REM ROUTINE CONTROLLO
105 FORK=1TOLEN(A$):A1$=MID$(A$,K,1)
110 E=-(A1$<" "ORA1$>"]"ANDA1$<"♠"ORA1$>"♦")
115 IFETHENP=K:RETURN
120 NEXT:RETURN
```

E' anche possibile, ogni volta che si incontra un carattere non valido, eliminarlo o sostituirlo con un determinato carattere, ad esempio, lo spazio. Ad esempio: eliminare i caratteri non alfanumerici o di interpunzione dalla stringa A$. Il dato di ingresso è in A$, il dato in uscita eventualmente corretto è in A$; E=1 se A$ è stata modificata, E=0 se no. Vedi al riguardo come abbiamo realizzato il programma ES5.

```
1 REM ES5
5 REM PROVA ROUTINE CONTROLLO
10 PRINT"♥STRINGA DA CONTROLLARE"
20 INPUTA$:GOSUB100
30 PRINT"STRINGA "MID$("NON CORRETTA",5-4*E1)
35 IFE1THENPRINT"QSTRINGA CORRETTA:":PRINTA$
40 INPUT"QQPREMI [RETURN]";I$
50 RUN
100 REM ROUTINE CONTROLLO
105 E1=0:K1=0
107 FORK=1TOLEN(A$):K1=K1+1:A1$=MID$(A$,K1,1)
110 E=-(A1$<" "ORA1$>"]"ANDA1$<"♠"ORA1$>"♦")
115 IF E THEN125
120 NEXT:RETURN
125 A$=LEFT$(A$,K1-1)+MID$(A$,K1+1)
126 E1=1:K1=K1-1:GOTO120
```

Si può modificare la linea 115 in modo che al posto del carattere non valido venga inserito il carattere “?”, come nel programma ES5.1.

```
1 REM ES5.1
5 REM PROVA ROUTINE CONTROLLO
10 PRINT"♥STRINGA DA CONTROLLARE"
20 INPUTA$:GOSUB100
30 PRINT"STRINGA "MID$("NON CORRETTA",5-4*E1)
35 IFE1THENPRINT"QSTRINGA CORRETTA:":PRINTA$
40 INPUT"QQPREMI [RETURN]";I$
50 RUN
100 REM ROUTINE CONTROLLO
105 E1=0:K1=0
107 FORK=1TOLEN(A$):K1=K1+1:A1$=MID$(A$,K1,1)
110 E=-(A1$<" "ORA1$>"]"ANDA1$<"♠"ORA1$>"♦")
115 IF E THEN125
120 NEXT:RETURN
125 A$=LEFT$(A$,K1-1)+"?"+MID$(A$,K1+1)
126 E1=1:GOTO120
```

TIPO

Se si effettua la INPUT di un dato numerico e si tenta di introdurre un dato alfanumerico, il sistema segnala "?REDO FROM START" e richiede il dato. Poichè non è possibile eliminare questo inconveniente che spesso porta a confusione sullo schermo, scrolling o altro, è preferibile effettuare la lettura con una variabile stringa e controllare la correttezza del dato da programma. Ciò provoca un rallentamento nell'esecuzione, ma evita altri inconvenienti. Per esempio: controllare se A$ contiene un numero intero senza segno. Il dato in ingresso è in A$: in uscita E=0 se va bene, E=1 se no. Segue il programma ES6.

```
1 REM ES6
5 REM PROVA ROUTINE CONTROLLO
10 PRINT"♥STRINGA DA CONTROLLARE"
20 INPUTA$:GOSUB100
30 PRINT"STRINGA "MID$("NON NUMERICA",5-4*E)
40 INPUT"QQPREMI [RETURN]";I$
50 RUN
100 REM ROUTINE CONTROLLO
105 E=0:FORK=1TOLEN(A$):A1$=MID$(A$,K,1)
110 E=-(A1$<"0"ORA1$>"9")
115 IFETHENK=LEN(A$)
120 NEXT:RETURN
```

E' possibile aggiungere il controllo sulla presenza del segno modificando le linee 100 e 105, come indicato nel programma ES7.

```
1 REM ES7
5 REM PROVA ROUTINE CONTROLLO
10 PRINT"♥STRINGA DA CONTROLLARE"
20 INPUTA$:GOSUB100
30 PRINT"STRINGA "MID$("NON NUMERICA",5-4*E)
40 INPUT"QQPREMI [RETURN]";I$
50 RUN
100 REM ROUTINE CONTROLLO
101 E=0:A=1-(LEFT$(A$,1)="+"ORLEFT$(A$,1)="-")
105 FORK=ATOLEN(A$):A1$=MID$(A$,K,1)
110 E=-(A1$<"0"ORA1$>"9")
115 IFETHENK=LEN(A$)
120 NEXT:RETURN
```

In questo modo, se il primo carattere della stringa numerica è + o -, il controllo di validità parte dal secondo carattere. E' possibile introdurre anche un controllo sulla presenza di un solo punto decimale in testa al numero oppure all'interno, utilizzando una variabile PD. Abbiamo realizzato come esempio il programma ES8.

```
1 REM ES8
5 REM PROVA ROUTINE CONTROLLO
10 PRINT"STRINGA DA CONTROLLARE"
20 INPUTA$:GOSUB100
30 PRINT"STRINGA "MID$("NON NUMERICA",5-4*E)
40 INPUT"PREMI [RETURN]";I$
50 RUN
100 REM ROUTINE CONTROLLO
101 E=0:A=1-(LEFT$(A$,1)="+"ORLEFT$(A$,1)="-")
103 PD=0
105 FORK=ATOLEN(A$):A1$=MID$(A$,K,1)
110 E=-(A1$<"0"ORA1$>"9")
112 IFEANDD(A1$=".")ANDPD=0THENPD=K:E=0
115 IFETHENK=LEN(A$)
120 NEXT
125 IFPD=LEN(A$)THENE=1
130 RETURN
```

Includere il controllo su un generico numero reale vorrebbe dire:

- cercare la presenza del carattere E,
- controllare che i caratteri seguenti rappresentino un intero con segno, compreso entro determinati limiti, ma ciò complica parecchio le cose.

VALORE

Questo controllo è fatto per accertarsi che il dato in ingresso sia compreso entro certi limiti. Per i dati numerici la cosa è abbastanza semplice:

200 IF VAL(A$) <"0" OR VAL(A$)>"24" THEN PRINT "NON VALIDO!" : GOTO 100

se il dato è letto come stringa; oppure:

200 IF A < 10 OR A > 100 THEN...

quando il dato è letto come numero.
Per i dati stringa bisogna ricordare che per l'ordinamento viene considerato il codice ASCII-CBM di ogni carattere e il numero dei caratteri della stringa; così:

"AIUTO " > "AIUTO"
"13" > "11"
"barca" < "Barca"

e il confronto può essere effettuato come:

210 IF A$ < B$ THEN...
oppure: 210 IF A$ > "11" THEN...

Il controllo B) viene effettuato con gli stessi controlli sul carattere già visti negli esempi, ma leggendo i caratteri singolarmente con un'istruzione GET (o una PEEK dalle locazioni di indirizzo noto); in questo caso è possibile non accettare un carattere non valido e proseguire.
Un problema legato all'introduzione dei dati carattere per carattere è la visualizzazione dei dati introdotti (feedback). Infatti, a differenza di INPUT, GET non visualizza nè un cursore lampeggiante nè tantomeno il carattere introdotto (la cosa non è automatica, il video e la tastiera sono due periferiche indipendenti); bisogna perciò provvedere da programma.
Per rimediare alla mancanza di feedback è spesso sufficiente un'istruzione che stampi il carattere una volta che questo sia considerato valido. Fanno eccezione:

- il carattere virgolette, per il quale bisogna avere particolare attenzione, perchè fa entrare e uscire dal QUOTE MODE anche in fase di stampa,
- i caratteri di controllo che eventualmente devono essere riconosciuti ed eseguiti.

Il problema di un cursore che visualizzi la posizione dove dovrà essere inserito il prossimo carattere può essere risolto in vari modi:

- se non è permesso tornare indietro sulla stringa appena letta è spesso sufficiente un qualunque carattere, magari in REVERSE o di colore differente, che indichi dove verrà aggiunto il prossimo carattere;
- se è permesso modificare la stringa riscrivendoci sopra si possono utilizzare tecniche del tipo:
  - cursore fisso, in campo inverso, che indica il carattere sul quale si trova;
  - cursore lampeggiante che mostra alternativamente il carattere sul quale si trova e/o lo stesso carattere in campo inverso o un altro carattere prefissato (di solito lo spazio o una lineetta);
  - cursore sottostante la stringa in lettura, che indica quale carattere verrà modificato o aggiunto premendo un tasto. A seconda delle applicazioni possono essere studiati altri tipi di presentazione del cursore.

Inoltre può essere utile controllare il colore dello sfondo (background) e quindi attribuire al cursore, e all'input visualizzato, un colore contrastante con esso.

Programma esempio ES9: legge una stringa solo alfanumerica, di lunghezza massima fissata, trasforma il carattere virgolette nel carattere apice. DEL cancella l'ultimo carattere, SHIFT- CLR/HOME cancella tutta la stringa. X e Y sono le coordinate di schermo dove iniziare l'INPUT (Y < 24); N è la lunghezza massima (N < 40), A$ è la stringa letta.

```
1 REM ES9
4 REM INPUT CONTROLLATO
5 B$="▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮"
6 S$="                                       "
7 C$="▄█▌"
10 N=20:X=0:Y=5:PRINT"▯"LEFT$(B$,Y+2)SPC(X);
15 FORK=1TON:PRINT"+";:NEXT:GOSUB100
20 STOP
100 REM ROUTINE CONTROLLO
105 A$="":PRINTLEFT$(B$,Y+1)SPC(X)LEFT$(S$,N)
110 PRINTLEFT$(B$,Y+1)SPC(X)C$;
115 FORK=0TO1:K=0
120 GETH$:IFH$=""THEN120
125 IFH$=CHR$(13)THENPRINT" █▌";:K=1:NEXT:RETURN
130 IFH$=CHR$(20)ANDLEN(A$)THEN155
135 IFH$=CHR$(147)THENK=1:PRINT" █▌";:NEXT:GOTO105
140 IFH$=CHR$(34)THENH$="'"
145 IFLEN(A$)<NTHEN160
150 NEXT
155 A$=LEFT$(A$,LEN(A$)-1):PRINT" ██▌"C$;:NEXT
156 GOTO135
160 IF(H$>=" "ANDH$<="["ORH$>"-"ANDH$<"+")THEN170
165 GOTO150
170 A$=A$+H$:PRINTH$C$;:GOTO150
```

Programma esempio ES10 : legge una stringa solo numerica, massimo N cifre totali con un punto decimale. Sono utilizzati i caratteri di controllo DEL, SHIFT-CLR/HOME e movimento cursore orizzontale.

```
1 REM ES10
4 REM INPUT CONTROLLATO
5 B$="▒QQQQQQQQQQQQQQQQQQQQQQQQ"
6 S$="                                        "
10 X=9:Y=10:N=10
15 PRINT"♥":GOSUB200
20 PRINT:PRINT:PRINT"NUMERO LETTO:"A$:STOP
200 REM ROUTINE CONTROLLO
205 PD=0:A$="":C=1
208 PRINTLEFT$(B$,Y+2)SPC(X)"▒"LEFT$(S$,N)"■";
210 PRINTLEFT$(B$,Y+1)SPC(X)LEFT$(S$,N+1)
215 FORK=0TO1
216 K=0:IFPDTHENPD=-PD*(MID$(A$,PD,1)=".")
220 PRINTLEFT$(B$,Y+1)SPC(X)LEFT$(A$,C-1);
222 PRINT"▒"MID$(A$+" ",C,1)"■"MID$(A$,C+1)" ";
225 PRINTLEFT$(B$,Y+1)SPC(X+C-1);
230 GETH$:IFH$=""THEN230
235 IFH$=CHR$(13)ANDPD<LEN(A$)THEN295
240 IFH$=CHR$(147)THENK=1:NEXT:GOTO205
245 IF C=(LEN(A$)+1)ANDLEN(A$)>=NTHEN260
250 IF(H$="."AND(PD=0))=0THEN255
253 PD=C:A$=LEFT$(A$,C-1-(C=0))+H$
254 A$=A$+MID$(A$,C+1-(C=0)):C=C+1:NEXT
255 IFH$<"0"ORH$>"9"THEN260
258 A$=LEFT$(A$,C-1-(C=0))+H$+MID$(A$,C+1-(C=0))
259 C=C+1:NEXT
260 IFH$=CHR$(29)THENC=C-(C<=LEN(A$)):NEXT
265 IFH$=CHR$(157)THENC=C+(C>0):NEXT
270 IFH$<>CHR$(20)THEN230
275 IFC=1ORLEN(A$)=0THENNEXT
280 PD=PD+PD*(C=(1+PD))+(C<=PD)
285 A$=LEFT$(A$,C-2)+MID$(A$,C):C=C-1
290 NEXT
295 PRINTMID$(A$+" ",C,1);:K=1:NEXT:RETURN
```

Come puoi notare il rendere utilizzabili più codici di controllo complica parecchio le cose costringendo a fare un gran numero di controlli (9 IF!). Il numero di controlli cresce anche aumentando la complessità del dato in ingresso; puoi provare a modificare l'ultima routine per accettare numeri espressi in forma esponenziale e te ne accorgerai.

L'utilizzo del cursore fisso in campo inverso costringe a ripetute PRINT, ogni volta che si modifica la posizione del cursore o la stringa letta. Vediamo, nel programma

ES11, come potrebbe essere la routine se si utilizzasse un cursore posto sotto la stringa di ingresso (la riga dove effettuare l'input non deve essere l'ultima in basso, cioè Y < 24).

```
1 REM ES11
4 REM INPUT CONTROLLATO
5 B$="SQQQQQQQQQQQQQQQQQQQQQQQQ"
6 S$="                             "
8 C$="▔▌▌":Q$=" ▌▌"
10 X=9:Y=10:N=10
15 PRINT"♥":GOSUB200
20 PRINT:PRINT:PRINT"NUMERO LETTO:"A$:STOP
200 REM ROUTINE CONTROLLO
205 PD=0:W=0
208 A$="":C=1
210 PRINTLEFT$(B$,Y+1)SPC(X)LEFT$(S$,N+1):PD=0
215 PRINTLEFT$(B$,Y+1)SPC(X)A$"  ";
218 IFPDTHENPD=PD+PD*(MID$(A$,PD,1)<>".")
220 PRINTLEFT$(B$,Y+2)SPC(X+C-1)C$;
225 GETH$:IFH$=""THEN225
230 IFH$=CHR$(13)ANDPD<LEN(A$)THENPRINTQ$;:RETURN
235 IFH$=CHR$(147)THENPRINTQ$;:GOTO205
240 IFC=LEN(A$)+1ANDLEN(A$)>=NTHEN260
245 IFH$="."AND(PD=0)THENPD=C:W=1
250 IFH$>"/"ANDH$<":"THENW=1
255 IFWTHEN290
260 IFH$=CHR$(29)THEN297
265 IFH$=CHR$(157)THENC=C+(C>1):PRINTQ$;:GOTO220
270 IFH$<>CHR$(20)THEN225
275 IFC<2ORLEN(A$)=0THEN215
280 PD=PD+PD*(PD+1=C)+(C<PD)
285 A$=LEFT$(A$,C-2)+MID$(A$,C):C=C-1:PRINTQ$;
286 GOTO215
290 W=0:A$=LEFT$(A$,C-1)+H$+MID$(A$,C+1)
295 C=C+1:PRINTQ$;:GOTO215
297 C=C-(C<LEN(A$)):PRINTQ$;:GOTO220
```

1.4.2 Maschere video

Questo termine indica l'uso di certe scritte fisse sul video, durante l'introduzione dei dati, che creano appunto una MASCHERA di guida alla scrittura dei dati. Il modo

più semplice per creare una maschera video è probabilmente quello di scrivere in colonna tutte le voci da richiedere; in fase di lettura ci si posizionerà dopo ognuna delle voci e si eseguirà la lettura.
Si possono complicare e rendere più estetiche le cose associando ad ognuna delle voci una posizione X,Y sullo schermo, alla quale effettuare la lettura del dato. Inoltre, se i dati in ingresso sono relativi a un argomento specifico è consigliabile ricordarlo a chi usa il programma con delle scritte appropriate. Rimandiamo per gli esempi ai programmi QUADRO1 e QUADRO2 del Paragrafo 3.5 del volume sul BASIC.

1.4.3 Presentazione dei dati

Riprendiamo qui l'argomento della presentazione dei dati sul video in modo ordinato (si veda il volume sul BASIC, che tratta l'argomento anche per la stampante).
Non avendo ancora a disposizione la grafica, per i dati numerici possiamo accontentarci di tabulati, e per dati più strutturati di maschere video.
Grazie alle funzioni TAB(X), che posiziona il cursore alla X-esima posizione a partire dall'inizio della linea corrente, SPC(X), che spazia di X caratteri SENZA SOVRASCRIVERE, e POS(X), che fornisce la posizione del cursore sulla linea LOGICA corrente (0 <=X <=79), è possibile incolonnare come si desidera i dati.
Vediamo qualche esempio.

TAVOLA PITAGORICA: è un esempio classico di incolonnamento di dati numerici, non tutti dello stesso numero di cifre. Il risultato è riportato nella Tabella 1.5.

TAVOLA PITAGORICA

	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30
4	4	8	12	16	20	24	28	32	36	40
5	5	10	15	20	25	30	35	40	45	50
6	6	12	18	24	30	36	42	48	54	60
7	7	14	21	28	35	42	49	56	63	70
8	8	16	24	32	40	48	56	64	72	80
9	9	18	27	36	45	54	63	72	81	90
10	10	20	30	40	50	60	70	80	90	100

Tabella 1.5 Tavola Pitagorica ottenuta con ES12

L'unico problema è quello di spaziare 3 caratteri se il numero è minore di 10, 2 caratteri se esso è maggiore di 10 e minore di 100, 1 carattere se esso è maggiore di 99. Un'espressione che dà tali valori in corrispondenza dei controlli indicati (K è il numero) è:
(1 - (K < 10) - (K < 100)).
Devi ricordare che un numero intero viene stampato preceduto dal segno - o da uno spazio e seguito da uno spazio.
Un programma per realizzare la tavola pitagorica è ES12; esso utilizza la funzione SPC.

```
1 REM ES12
5 REM TAVOLA PITAGORICA
10 PRINT"■  TAVOLA  PITAGORICA";
15 PRINTSPC(22)"------------------"
20 FORK=1TO10
25 PRINT"■"SPC(1-(K<10));K;:NEXTK:PRINT
30 FORK=1TO10
35 PRINT"■"SPC(-(K<10));K;SPC(1);
40 FORJ=2TO10
45 A=K*J:PRINT"■"SPC(-(A<100)-(A<10));A;
50 NEXTJ:PRINT"■";:NEXTK
60 PRINT"■";
70 GOTO70
```

Per utilizzare la funzione TAB si devono fare delle modifiche al programma e fare attenzione al fatto che: la funzione TAB non agisce sul video se l'argomento definisce una posizione già superata dal cursore.

LANCIO DI DUE DADI: il programma ES14 visualizza i dati relativi ad una simulazione del lancio di due dadi non truccati (per truccarli, basta modificare la funzione definita alla linea 30), ovvero il numero delle estrazioni avvenute e previste di ognuna delle 11 possibili somme. Dopo ogni estrazione i dati vengono aggiornati.

```
1 REM ES14
5 REM LANCIO DADI
7 PRINT"■";
10 INPUT"NUMERO LANCI (L%>0 & L%<=1000)";L%
15 IFL%<1ORL%>1000THEN10
25 DIMFO(10),FP(10)
```

```
30 DEFFND(X)=INT(RND(0)*6)+1
35 FORK=0TO10
36 FP(K)=(6-ABS(5-K))/36:NEXTK
40 FORJ=1TOL%:A=FND(0)+FND(0)-2:FO(A)=FO(A)+1
47 PRINT"□ SOMMA"SPC(10)"ESTRAZIONI"SPC(16);
48 PRINT"DADI"SPC(5)"OSSERVATE   PREVISTE":PRINT
50 FORK=0TO10
51 PRINTTAB(1-(K<8))K+2;
53 X=FO(K)
54 PRINTTAB(12-(X<1000)-(X<100)-(X<10));FO(K);
55 X=INT(FP(K)*J*100+.5)/100
57 PRINTTAB(22-(X<1000)-(X<100)-(X<10)-(X<1))X
60 NEXTK
65 PRINT"■N. LANCI: ";
70 PRINTSPC(2-(J<1000)-(J<100)-(J<10))J"/"L%;
75 FORX=1TO400:NEXTX:NEXTJ
```

TIPI DI INCOLONNAMENTO: può essere studiato un incolonnamento adatto ai tipi di dati da visualizzare. Nell'esempio che segue i dati sono letti da linee DATA nel programma, in particolare i numeri reali sono scelti in modo da non essere trasformati nella corrispondente forma esponenziale (per non complicare i controlli). Abbiamo realizzato il programma ES15 come esempio.

```
1 REM ES15
5 REM INCOLONNAMENTI
7 S$="00000":RESTORE
10 PRINT"□■ALFABETICO---INTERO-----DECIMALE"
15 PRINT:FORK=1TO8:READA$,A,B:GOSUB200:GOSUB300
20 PRINTTAB(1)A$;TAB(8)SPC(11-LEN(A1$));A1$;
23 PRINTSPC(8-LEN(B1$))B1$".";B2$
25 NEXTK
30 END
200 REM ROUTINE FORMATO INTERI
203 A1$=MID$(STR$(A),2)
205 IFLEN(A1$)>3THEN220
210 IFLEN(A1$)>7THEN230
215 RETURN
220 X1$=LEFT$(A1$,LEN(A1$)-3)+"."
223 A1$=X1$+MID$(A1$,LEN(A1$)-2)
```

```
225 GOTO210
230 X1$=LEFT$(A1$,LEN(A1$)-7)+"."
233 A1$=X1$+MID$(A1$,LEN(A1$)-6):RETURN
300 B1$=MID$(STR$(INT(B)),2)
305 B2$=MID$(STR$(B-VAL(B1$)),3)
307 B2$=LEFT$(B2$+S$,5)
310 RETURN
1000 DATAAAA,38450,26.707,BBBBB,4900,2.404
1005 DATACCCC,400002,310.02,DDD,256000,443.2
1010 DATAEEEEE,2000,20,FFF,100,80.465
1015 DATAGG,10000000,0,HHHHH,7860,235.796
```

LETTURA DALLA MEMORIA (MEMORY DUMP): il problema di tabulare è più semplice se i dati hanno una lunghezza nota e costante. Nell'esempio che segue, dati un indirizzo iniziale e uno finale di memoria, viene presentato il contenuto della memoria 8 byte per linea, con il seguente formato:

-XXXXX-AAAAAAAA-YY-YY-YY-YY-YY-YY-YY-YY-

dove:

- XXXXX è l'indirizzo del primo degli 8 byte,
- ogni A rappresenta il carattere ASCII stampabile rappresentato nel byte o un punto,
- le 8 coppie YY rappresentano il contenuto degli 8 byte espresso in esadecimale.

Il programma ES16 realizza una tabulazione.

```
1 REM ES16
5 REM MEMORY DUMP
10 H$="0123456789ABCDEF"
20 INPUT"INDIRIZZO INIZ.: ";A1
25 IF0>A1ORA1>65534THEN20
30 INPUT"INDIRIZZO FIN.: ";A2
35 IFA1>A2ORA2>65535THEN30
40 INPUT"OK ";R$:IFR$="N"THEN20
50 IFR$<>"S"THEN40
60 L=1:K=A1
70 K$=RIGHT$("0000"+MID$(STR$(K),2),5)
80 PRINTSPC(1)K$SPC(1)"■";
95 FORJ=0TO7:A=PEEK(K+J)
100 A$=".":IFA>31ANDA<128ORA>159THENA$=CHR$(A)
120 PRINTA$;
```

```
125 POKE212,0
130 NEXTJ:PRINT"■";
145 FORJ=0TO7:A=PEEK(K+J)
147 X1=(AAND240)/16+1:X2=(AAND15)+1
150 PRINT"▮▮"MID$(H$,X1,1)MID$(H$,X2,1);
160 NEXTJ:PRINT
170 L=L+1+(L=24)*24:IFL<>1THEN190
180 GETR$:IFR$<>CHR$(13)THEN180
190 K=K+8:IFK<=A2THEN70
200 GOTO20
```

Osserva la semplice tecnica usata per contare le linee stampate, in modo da interrompere la stampa (fino alla pressione di CR), evitando lo scrolling indesiderato.

GRAFICO DELLA FUNZIONE SIN(X): il programma ES17 è un piccolo esempio di grafica con i caratteri. Il grafico compare linea per linea, sfruttando lo scrolling dello schermo.

```
1 REM ES17
5 REMGRAFICO FUNZIONE SENO
7 P$=".▮▮":A$="*▮▮"
10 PRINT"□"SPC(250)SPC(250)SPC(250)SPC(250);
15 PRINTTAB(13)"FUNZIONE  SENO"SPC(13);
20 FORK=1TO40:PRINT"=";:NEXTK
25 PRINT"ANG. -1---------------0---------------1"
30 FORK=0TO360STEP15
35 PRINTRIGHT$("  "+STR$(K),3)"▮▮▮▮";
40 T=22+SIN(K/180*π)*16
45 IFT>6THENPRINTTAB(6)P$;
50 IFT<22THEN80
55 IFT=22THENPRINTTAB(T)A$TAB(38)P$;:GOTO65
60 PRINTTAB(22)" ▮▮"TAB(T)A$;
63 IFT<38THENPRINTTAB(38)P$;
65 PRINT:NEXTK
70 GOTO70
80 PRINTTAB(T)A$TAB(22)"|▮▮"TAB(38)P$;:GOTO65
```

1.5 GRAFICA IN MODO CARATTERI

Ben 128 dei 256 caratteri visualizzabili sullo schermo (74 nel set maiuscolo/minuscolo) non sono nè alfanumerici nè simbolici, ma CARATTERI GRAFICI. Il loro uso appropriato può permettere di creare disegni e grafici di discreta qualità. Ci occupiamo qui di questo argomento, dopo aver chiarito il significato di alcuni termini.

- PIXEL: è la più piccola unità di informazione grafica visualizzabile sullo schermo. Comunemente si usa dire PUNTO, ma, talvolta, un PIXEL può essere composto da diversi PUNTI DELLO SCHERMO, il cui numero indica la DIMENSIONE DEL PIXEL.
- RISOLUZIONE: è il numero massimo di PIXEL visualizzabili sullo schermo.
- NUMERO COLORI: è il numero di colori disponibili, accompagnato da quello del massimo numero di colori diversi utilizzabili in un certo raggruppamento di punti.
- FORMA DEL PIXEL: non è detto che il PIXEL sia per forza rettangolare, quadrato o puntiforme (alcuni calcolatori visualizzano PIXEL di forma esagonale).

1.5.1 Grafica a bassissima risoluzione

In modo caratteri lo schermo è suddiviso in 25 linee di 40 caratteri ciascuna. Se utilizziamo un carattere come PIXEL, possiamo ottenere grafici con le seguenti caratteristiche:

- RISOLUZIONE: 25*40=1000 PIXEL
- NUM. COLORI: 1 su 16 per ogni PIXEL
- FORMA DEL PIXEL: RETTANGOLARE o carattere ASCII-CBM
- DIMENSIONI DEL PIXEL: 8*8=64 PUNTI dello schermo (1 carattere).

Per fare comparire un pixel in una determinata posizione è sufficiente una PRINT o una coppia di POKE (meglio perchè evitano lo scrolling) basate sulle coordinate della posizione. A questo scopo puoi scegliere di riferirti allo schermo in uno dei sistemi di coordinate riportati in Figura 1.3.

Quelli più utilizzati sono a) e b), e il secondo dei due è forse il più naturale per chi è abituato a tracciare grafici, in quanto coincide con il primo quadrante in coordinate cartesiane.

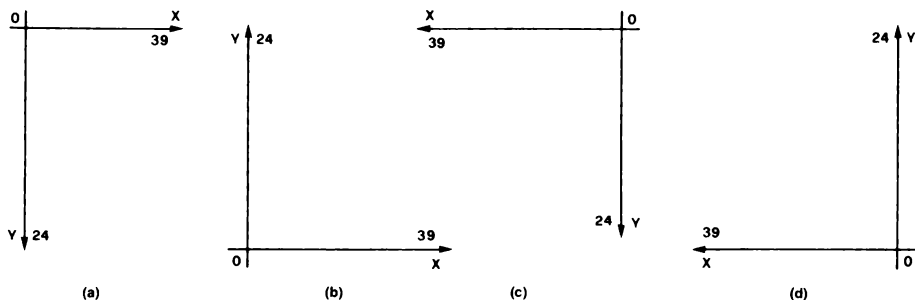



Figura 1.3 Sistemi di coordinate

Per le coordinate i limiti sono:
0 < =X < =39 e 0 < =Y < =24.
Una routine per disegnare un pixel sullo schermo può essere allora realizzata come nel programma esempio ES18.

```
1 REM ES18
10 REM PROVA ROUTINE
15 BV=1024:BC=55296:CH=160:CO=8:SC=13
20 PRINT"♥";:INPUT "COORDINATA X=";X
25 INPUT "COORDINATA Y=";Y:GOSUB1000
30 GOTO30
1000 REM ROUTINE DISEGNA PUNTO
1005 IFX<0ORY<0ORX>39ORY>24THEN1025
1010 A=40*(24-INT(Y))+INT(X)
1015 POKEBV+A,CH:POKEBC+A,CO
1020 RETURN
1025 PRINT"?FUORI DAL VIDEO","ERRORE":END
```

dove:
X,Y sono le coordinate del pixel nel sistema b),
BV è l'indirizzo base del video (1024 di norma),
CO è L'indirizzo della mappa colore (55296),
CH è il codice del carattere che si vuole usare come PIXEL (160 è il più comune),

CL è il colore del PIXEL (compreso tra 0 e 15).
Alla linea 1005 viene fatto un controllo sulla validità delle coordinate e in caso esse non siano nello schermo viene segnalato un errore e il programma si ferma. Puoi aggiungere controlli per i valori di CH e di CL per evitare il messaggio di errore: ?ILLEGAL QUANTITY ERROR, come nel programma ES19. Sulla cassetta trovi anche il programma ES19.1 che contiene una variazione rispetto ad ES19.

```
1 REM ES19
10 REM PROVA ROUTINE
15 BV=1024:BC=55296:CH=160:CO=8:SC=13
20 PRINT"J";:INPUT "COORDINATA X=";X
25 INPUT "COORDINATA Y=";Y:GOSUB1000
30 GOTO30
1000 REM ROUTINE DISEGNA PUNTO CORRETTO
1005 XX=X:YY=Y
1010 IFXX>0ANDXX<40THEN1017
1015 XX=XX-(XX<0)*XX+(XX>39)*(XX-39):GOTO1010
1017 IFYY>0ANDYY<24THEN1025
1020 YY=YY-(Y=0)*YY+(YY>23)*(YY-23):GOTO1017
1025 A=40*(24-INT(YY))+INT(XX)
1030 POKEBV+A,CH:POKEBC+A,CO
1035 RETURN
```

Se sei assolutamente sicuro che i valori di X e Y non siano mai fuori dai limiti puoi eliminare la linea 1005.
Per cancellare un pixel puoi o ridisegnarlo nel colore dello sfondo o usare CH=32 e riscriverci sopra.
Per vedere un altro esempio di quanto è stato detto puoi provare il programma ES20.

```
1 REM ES20
5 REM GRAFICO DI UNA FUNZIONE
7 MX=39:MY=24:CC=646:CH=160
10 POKE53280,0:POKE53281,0:POKECC,1
12 DIMV(MX)
15 BV=1024:BC=55296:CO=2:KB=630:R$=CHR$(13)
40 F$="6"
45 DEF FNA(X)=6
```

```
 50 PRINT"■HAI GIA' INSERITO LA FUNZIONE"
 55 INPUT I$:IFI$="S"ORI$="SI"THEN100
 60 INPUT"FUNZIONE:";F$:IFF$=""THEN60
 65 IFLEN(F$)>65THENPRINT"TROPPO LUNGA":GOTO60
 70 POKECC,0:PRINT"■■■40 F$="CHR$(34)F$CHR$(34)
 75 PRINT"45 DEF FNA(X)="F$R$"RUN■";
 80 E$=R$+R$+R$:GOSUB1500:END
 100 REM
 200 INPUT"■ESTREMI (X1,X2):";X1,X2
 205 IF X1>=X2 THEN200
 210 PX=(X2-X1)/MX:E$="G■500"+R$
 215 PRINT"■STO C A L C O L A N D O !"
 220 GOSUB1500:MI=FNA(X1):MA=MI
 225 FORK=0TOMX:A=FNA(X1+K*PX):V(K)=A
 230 IFA>MATHENMA=A:GOTO240
 235 IFA<MITHENMI=A
 240 NEXT
 245 PY=(MA-MI)/MY
 246 IFPY=0THENSTOP
 250 PRINT"■PREPARAZIONE G R A F I C O !"
 255 FORK=0TOMX:V(K)=(V(K)-MI)/PY:NEXT
 260 PRINT"■";:FORK=0TOMX:X=K:Y=V(K):GOSUB1000
 265 NEXT:GOSUB1520
 268 GETA$:IFA$=""THEN268
 270 PRINT"*** "F$" ***";:GOSUB1520
 275 GETA$:IFA$=""THEN275
 280 PRINT"■■ 1 - TORNA AL GRAFICO"
 282 PRINT" 2 - CAMBIA GLI ESTREMI"
 284 PRINT" 3 - CAMBIA LA FUNZIONE"
 298 PRINT" 0 - TERMINA IL PROGRAMMA"
 300 INPUT"■   - COSA SCEGLI";I$
 305 IFI$<"0"ORI$>"3"THEN280
 310 ON VAL(I$)GOTO260,200,50
 315 END
 500 PRINT"■ATTENZIONE LA FUNZIONE NON E'"
 505 PRINT"DEFINITA CORRETTAMENTE"
 510 PRINT"NELL'INTERVALLO:"X1"-"X2
 580 PRINT"PREMI [F1] PER RIDEFINIRLA"
 585 GOSUB1520
 590 GETA$:IFA$<>"■"THEN590
 595 GOTO50
 999 STOP
```

```
1000 REM DISEGNA PUNTO
1005 IFX<0ORY<0ORINT(X)>39ORINT(Y)>24THEN1050
1010 A=40*(24-INT(Y))+INT(X)
1015 POKEBV+A,CH:POKEBC+A,CO
1020 RETURN
1050 PRINT"?FUORI DAL VIDEO","ERRORE":END
1500 REM ON ERROR GOTO
1501 REM METTE IN ON
1505 FORK=1TOLEN(E$)
1510 POKEKB+K,ASC(MID$(E$,K,1)):NEXT
1515 POKE198,LEN(E$):RETURN
1518 REM METTE IN OFF
1520 POKE198,0:RETURN
```

Nel TRACCIARE LINEE o altri grafici, se le coordinate di schermo di due punti differiscono per più di un'unità, i due punti appaiono separati. Se desideri vedere i punti uniti, devi tracciare linee da un punto di coordinate X1,Y1 a un punto di coordinate X2,Y2, con differenza di coordinate minore di uno.

Si possono presentare 4 casi:

a) I due punti si trovano sulla stessa linea orizzontale, cioè Y1=Y2; una routine può essere quella riportata nell'esempio ES22.

```
1 REM ES22
7 BV=1024:BC=55296:CH=160:CO=8
10 INPUT"X1= ";X1
20 INPUT"X2= ";X2
25 INPUT"Y1= ";Y1
27 Y2=Y1
30 PRINT"":GOSUB1050
40 GOTO40
1050 REM LINEA ORIZZONTALE: Y1=Y2
1055 IFX1*Y1<0ORX2*Y2<0THENSTOP
1057 IFX1>39ORY1>24ORX2>39ORY2>24THENSTOP
1060 IFX1>X2THENT=X1:X1=X2:X2=T
1065 A=40*(24-Y1)+X1-1
1070 FORK=1TO(X2-X1+1):POKEBV+A+K,CH
1073 POKEBC+A+K,CO:NEXTK
1075 RETURN
```

b) I due punti si trovano sulla stessa linea verticale, cioè X1=X2; come nel programma esempio ES23.

```
1 REM ES23
7 BV=1024:BC=55296:CH=160:CO=8
10 INPUT"X1= ";X1
20 INPUT"Y1= ";Y1
25 INPUT"Y2= ";Y2
27 X2=X1
30 PRINT"J":GOSUB1050
40 GOTO40
1050 REM LINEA VERTICALE: X1=X2
1055 IFX1*Y1<0ORX2*Y2<0THENSTOP
1057 IFX1>39ORY1>24ORX2>39ORY2>24THENSTOP
1060 IFY1>Y2THENT=Y1:Y1=Y2:Y2=T
1065 A=40*(24-Y1)+X1-1
1070 FORK=0TO(Y2-Y1):POKEBV+A-K*40,CH
1073 POKEBC+A-K*40,CO:NEXTK
1075 RETURN
```

c) I due punti si trovano su una retta a 45 gradi, cioè ABS(X2-X1)=ABS(Y2-Y1), come nel programma esempio ES24.

```
1 REM ES24
7 BV=1024:BC=55296:CH=160:CO=8
10 INPUT"X1= ";X1
20 INPUT"X2= ";X2
25 INPUT"Y1= ";Y1
27 INPUT"Y2= ";Y2
30 PRINT"J":GOSUB1050
40 GOTO40
165 IFABS(X2-X1)<>ABS(Y2-Y1)THENSTOP
1000 REM ROUTINE DISEGNA PUNTO
1005 IFX<0ORY<0ORX>39ORY>24THEN1025
1010 A=40*(24-INT(Y))+INT(X)
1015 POKEBV+A,CH:POKEBC+A,CO
1020 RETURN
1025 PRINT"?FUORI DAL VIDEO","ERRORE":END
1050 REM LINEA A 45 GRADI:ABS(X2-X1)=ABS(Y2-Y1)
```

```
1055 IFX1*Y1<0ORX2*Y2<0THENSTOP
1057 IFX1>39ORY1>24ORX2>39ORY2>24THENSTOP
1070 X=X1:FORK=Y1TOY2STEPSGN(Y2-Y1):Y=K
1073 GOSUB1005:X=X+1:NEXTK
1075 RETURN
```

d) I due punti non sono allineati nei modi precedenti.

Sulla cassetta trovi registrati anche i 3 programmi ES22.1, ES23.1 e ES24.1, che sono variazioni dei programmi precedenti.
Il caso d) è il più comune e il più difficile da risolvere. Infatti non basta tracciare una linea, ma si cerca di tracciare una buona linea, cioè una linea con le seguenti caratteristiche:

- inizio e fine nei punti dati,
- il più possibile diritta,
- a densità costante,
- senza discontinuità.

In sintesi il problema consiste nel trovare il numero di punti necessario per tracciare la linea, tenendo conto della risoluzione e dell'approssimazione ottenibile. Il calcolo più semplice consiste nello scegliere come numero dei punti il maggiore tra i valori ABS(X2-X1) e ABS(Y2-Y1), dividere entrambi gli intervalli per questo numero per ricavare l'incremento per ogni coordinata (sarà 1 o minore di 1) e tracciare la linea punto per punto.

```
1 REM ES25
5 BV=1024:BC=55296:CH=160:C0=8
10 PRINT"♥COORDINATE DUE PUNTI"
15 INPUT"COORDINATA X1=";X1
16 IFX1<0ORX1>39THEN15
20 INPUT"COORDINATA Y1=";Y1
21 IFY1<0ORY1>23THEN20
25 INPUT"COORDINATA X2=";X2
26 IFX2<0ORX2>39THEN25
30 INPUT"COORDINATA Y2=";Y2
31 IFY2<0ORY2>23THEN30
40 GOSUB1200
50 GOTO50
1000 REM PLOT
1010 A=40*(24-INT(Y))+INT(X)
```

```
1015 POKEBV+A,CH:POKEBC+A,CO
1020 RETURN
1200 REM DRAW DA X1,Y1 A X2,Y2
1205 L=ABS(X2-X1)
1206 IF ABS(Y2-Y1)>L THEN L=ABS(Y2-Y1)
1210 DX=(X2-X1)/L:DY=(Y2-Y1)/L
1215 X=X1+.5:Y=Y1+.5
1220 FORK=1TOL:GOSUB1000:X=X+DX:Y=Y+DY:NEXTK
1225 RETURN
```

Il grafico tracciato con il programma ES25 presenta molte irregolarità, che dipendono dalle dimensioni del pixel.
Il programma ES25.1 disegna una linea sovrapponendola a una griglia.

```
1 REM ES25.1
5 BV=1024:BC=55296:CH=160:C0=8
20 X1=4:Y1=13:X2=10:Y2=21:PRINT"♥";
25 PRINT"█████";
30 FORK=1TO10:PRINT"█+++++++++++"SPC(30);:NEXT
40 GOSUB1200
50 GOTO50
1000 REM PLOT
1010 A=40*(24-INT(Y))+INT(X)
1015 POKEBV+A,CH:POKEBC+A,CO
1020 RETURN
1200 REM DRAW DA X1,Y1 A X2,Y2
1205 L=ABS(X2-X1)
1206 IF ABS(Y2-Y1)>L THEN L=ABS(Y2-Y1)
1210 DX=(X2-X1)/L:DY=(Y2-Y1)/L
1215 X=X1+.5:Y=Y1+.5
1220 FORK=1TOL:GOSUB1000:X=X+DX:Y=Y+DY:NEXTK
1225 RETURN
```

Seguono alcuni programmi esempio: ES26, ES27, ES28.

```
1 REM ES26
10 REM SPIRALE
20 BV=1024:BC=55296:CH=81:CO=8
30 X1=20:Y1=12:C=0:PRINT"♥";
40 FORJ=1TO25:X2=X1+J*COS(C*π/2)
```

```
50 Y2=Y1+J*SIN(C*π/2):GOSUB1200
60 C=C+1+4*(C=3):X1=X2:Y1=Y2
70 NEXT
80 GOTO80
1000 REM PUNTO
1005 IFX<0ORY<0ORINT(X)>39ORINT(Y)>24THEN1050
1010 A=40*(24-INT(Y))+INT(X)
1015 POKEBV+A,CH:POKEBC+A,CO
1020 RETURN
1050 PRINT"?OUT OF SCREEN","ERROR":END
1200 REM DISEGNA
1205 L=ABS(X2-X1)
1207 IF ABS(Y2-Y1)>L THEN L=ABS(Y2-Y1)
1210 DX=(X2-X1)/L:DY=(Y2-Y1)/L
1215 X=X1+.5:Y=Y1+.5
1220 FORK=1TOL:GOSUB1000:X=X+DX
1223 Y=Y+DY:NEXTK:GOSUB1000
1225 RETURN
```

```
1 REM ES27
5 REM POLIGONO
10 BV=1024:BC=55296:CH=160
20 INPUT"▌NUM LATI (>0) ";NL:IFNL<1THEN20
30 INPUT"▓▓LUNGHEZZA(>0) ";LE:IFLE<0THEN30
40 Y1=0:X1=20-LE/2:GOSUB650
50 INPUT"▓▓▓ANCORA   (S/N) ";R$
55 IFR$="S"THEN20
60 IFR$<>"N"THEN50
70 END
650 REM POLIGONI
655 AN=0:FORJ=1TONL
660 X2=X1+LE*COS(AN):Y2=Y1+LE*SIN(AN)
665 GOSUB1200
670 X1=X2:Y1=Y2:AN=AN+2*π/NL
675 NEXT:RETURN
1000 REM PUNTI
1005 IFX<0ORY<0ORINT(X)>39ORINT(Y)>24THEN1050
1010 A=40*(24-INT(Y))+INT(X)
```

```
1015 POKEBV+A,CH:POKEBC+A,CO
1020 RETURN
1050 PRINT"?OUT OF SCREEN","ERROR":END
1200 REM LINEE
1205 L=ABS(X2-X1)
1206 IF ABS(Y2-Y1)>L THEN L=ABS(Y2-Y1)
1210 DX=(X2-X1)/L:DY=(Y2-Y1)/L
1215 X=X1+.5:Y=Y1+.5
1220 FORK=1TOL:GOSUB1000:X=X+DX
1223 Y=Y+DY:NEXTK:GOSUB1000
1225 RETURN
```

```
 1 REM ES28
 5 REM CIRCOLI
 10 BV=1024:BC=55296:CH=160:CO=6:XC=20:YC=12
 15 INPUT"RAGGIO (1-12) ";R:IFR<1ORR>12THEN15
 20 INPUT"COLORE (0-15) ";CO
 23 IFCO<0ORCO>15THEN20
 25 PRINT"□";:GOSUB600
 30 INPUT"■ANCORA (S/N)";R$:IFR$="S"THEN15
 35 IFR$<>"N"THEN30
 40 STOP
 600 REM CIRCOLO
 605 FORT=0.075TO2*πSTEP.075
 610 X=XC+R*SIN(T):Y=YC+R*COS(T)
 615 GOSUB1000:NEXT:RETURN
 1000 REM PUNTI
 1005 IFX<0ORY<0ORINT(X)>39ORINT(Y)>24THEN1050
 1010 A=40*(24-INT(Y))+INT(X)
 1015 POKEBV+A,CH:POKEBC+A,CO
 1020 RETURN
 1050 PRINT"?OUT OF SCREEN","ERROR":END
```

Sulla cassetta è registrato anche il programma ES27.1, una variazione di ES27.

1.5.2 Grafica a bassa risoluzione

Dei 74 caratteri grafici comuni ai due set di caratteri del COMMODORE 64 ce ne sono 16 che possono essere utilizzati per moltiplicare per 4 la risoluzione dello

schermo. In questo modo ogni pixel sarà un quarto di un carattere. Si ha:

- RISOLUZIONE: 50*80=4000 pixel,
- NUM. COLORI: 1 su 16 per ogni gruppo di 4 pixel,
- FORMA DEL PIXEL: rettangolare,
- DIMENSIONI DEL PIXEL: 4*4 punti (1/4 di carattere).

I 16 caratteri utilizzati sono riportati nella Figura 1.4; essi mostrano le 16 possibili combinazioni di punti per ogni carattere.

Figura 1.4 Caratteri per GRAFICA a BASSA RISOLUZIONE

La procedura per disegnare un pixel differisce da quella per la grafica a bassissima risoluzione in due aspetti in particolare:

A) per poter disegnare un pixel bisogna sapere se ci sono altri pixel nella zona carattere (che può contenere 4 pixel), dove esso dovrà comparire;

B) il carattere da scrivere dipende sia dalla posizione del pixel da disegnare, sia da quello che esiste già sullo schermo.

Per risolvere il problema A) è sufficiente, una volta calcolato l'indirizzo del carattere sullo schermo che conterrà il PIXEL da disegnare, confrontare il D-CODE del carattere, già presente in quella locazione, con i 16 riportati nella Figura 1.4. Se il confronto è positivo ci sono già pixel disegnati nella zona carattere, altrimenti il carattere eventualmente presente è alfanumerico o grafico e allora si può scegliere se sovrascrivere oppure non disegnare il pixel. La routine che controlla se c'è già qualche pixel presente sullo schermo, può essere come indicato in ES31, da 1250 a 1265.

```
1 REM ES31
3 REM CANCELLAZIONE PUNTI
5 DIMD(15),X(100),Y(100)
10 RESTORE:FORK=0TO15:READD(K):NEXT
11 DATA32,123,108,98,126,97,127,252,124
12 DATA255,225,254,226,236,251,160
15 BV=1024:BC=55296:CO=0:NP=0:PRINT"▯";
20 Q=INT(RND(1)*100)+3
30 FORJ=1TO100:X(J)=INT(RND(Q)*80)
35 Y(J)=INT(RND(Q)*50):NEXT
40 FORJ=1TO100:X=X(J):Y=Y(J):GOSUB1300:NEXT
45 FORJ=1TO100:X=X(J):Y=Y(J):GOSUB1350:NEXT
50 GOTO20
1250 REM SITUAZIONE POSIZIONE CARATTERE
1255 P1=PEEK(BV+A):H=16
1260 FORK=0TO15:IFP1=D(K)THENH=K:K=15
1265 NEXT:RETURN
1300 REM PUNTI
1305 IFX<0ORY<0ORINT(X)>79ORINT(Y)>49THEN1340
1310 A=40*(24-INT(Y/2))+INT(X/2)
1315 GOSUB1250:IFH=16THENH=0:IFNPTHENRETURN
1320 CH=D(H OR (2↑(X AND 1)*4↑(Y AND 1)))
1325 POKEBV+A,CH:POKEBC+A,CO
1330 RETURN
1340 PRINT"?OUT OF SCREEN  ERROR":END:RETURN
1350 REM CANCELLA PUNTI
1355 IFX<0ORY<0ORINT(X)>79ORINT(Y)>49THEN1340
1360 A=40*(24-INT(Y/2))+INT(X/2)
1365 GOSUB1250:IFH=16THENH=0:IFNPTHENRETURN
1370 CH=D(HAND(15-(2↑(X AND 1)*4↑(Y AND 1))))
1375 POKEBV+A,CH:POKEBC+A,CO
1380 RETURN
```

Il vettore D è stato inizializzato nelle prime linee del programma, da 5 a 12, usando READ e DATA.

Nel programma ES31 A è l'indirizzo della posizione carattere del video dove dovrà comparire il pixel, H contiene 16 se il carattere della A-esima posizione dello schermo non è compreso tra quelli indicati per questo tipo di grafica, un numero tra 0 e 15 se il carattere è, invece, tra quelli.

I valori limite per le coordinate sono:

$0 <= X <= 79$ e $0 <= Y <= 49$
e, utilizzando i sistemi di coordinate prima descritti, l'indirizzo della mappa video del carattere da modificare si calcola come:
A=40*INT(Y/2)+INT(X/2) nel sistema a), oppure:
A=40*(24-INT(Y/2))+INT(X/2) nel sistema b).
Il pixel dovrà comparire in una delle 4 posizioni possibili, e, a seconda del carattere letto dallo schermo, si dovrà ottenere il codice del carattere da riscrivere (sovrapporre).
Per risolvere il problema B), dopo aver chiamato la routine contenuta in ES31 nelle linee 1250-1265, si può calcolare CH come alla linea 1320 di ES31.
Questo perchè i dati nella matrice D sono ordinati in modo che il carattere nella posizione $0 <= N <= 15$ risulti la composizione dei 4 caratteri base ai quali sono associati i valori indicati in Figura 1.5.

VALORE IN D(N)	CARATTERE	N
D(1)=123		1
D(2)=108		2
D(4)=126		4
D(8)=124		8

Figura 1.5 Caratteri base per GRAFICA a BASSA RISOLUZIONE con coordinate di tipo (b)

Ad esempio i caratteri, il cui codice è in D(13) o in D(9), sono indicati in Figura 1.6.

VALORE IN D(N)	CARATTERE	N	COMPONENTI
D(13)=236		13=8+4+1	
D(9)=255		9=8+1	

Figura 1.6 Esempi di caratteri del vettore D

La routine inserita in ES31, da 1300 a 1340, serve per disegnare un pixel nella posizione X,Y; se NP $<>$ 0 non sovrascrive, se sullo schermo non esiste un carattere grafico, se NP=0 disegna comunque il pixel. Questo sottoprogramma è una variazione dell'analogo per la bassissima risoluzione.

Per preparare la routine che permette di CANCELLARE un pixel, come puoi vedere da ES31, è sufficiente copiare le linee dalla 1300 alla 1340 nelle linee dalla 1350 alla 1380 modificando la linea 1320, che diventa la 1370.

Sulla cassetta sono registrati due programmi che tracciano linee in bassa risoluzione, ES32 e ES33, che non sono qui listati.

ATTENZIONE: se utilizzi il sistema di coordinate che pone il punto 0,0 nell'angolo in alto a sinistra, devi riordinare i valori nelle linee DATA, in modo che ai 4 caratteri base siano associati i valori indicati in Figura 1.7.

```
VALORE IN D(N)   CARATTERE   N

  D(1)=126           ▘        1
  D(2)=124           ▝        2
  D(4)=123           ▖        4
  D(8)=108           ▗        8
```

Figura 1.7 Caratteri base per GRAFICA a BASSA RISOLUZIONE con coordinate di tipo (a)

Per tracciare linee in bassa risoluzione non fa molta differenza se i punti sono o meno allineati, proprio per il problema di dover controllare se ci sono già dei punti sullo schermo. L'unica modifica da apportare alla routine che disegna dell'esempio ES25 è in questo caso:

1220 FOR K=1 TO L:GOSUB 1300:X=X+DX:Y=Y+DY:NEXT K

Ora puoi adattare gli esempi riportati per la bassissima risoluzione alla bassa risoluzione. L'unico problema è che se disegni un pixel di un certo colore, cambi il colore degli altri tre pixel che si trovano nella stessa posizione carattere. E' difficile trovare una soluzione soddisfacente.

Puoi provare anche il programma ES34 che segue, che fa rimbalzare un pixel su tutto il video.

```
1 REM ES34
5 DIMD(15),X(100),Y(100)
10 RESTORE:FORK=0TO15:READD(K):NEXT
11 DATA32,123,108,98,126,97,127,252,124,255
12 DATA225,254,226,236,251,160
15 BV=1024:BC=55296:CO=6:NP=0:PRINT"□";
20 MX=79:MY=49:POKE53281,1
100 XN=INT(RND(0)*20)+20
105 YN=INT(RND(0)*10)+15
110 DX=1:DY=-1
```

```
120 FORF=0TO1:F=0
125 XO=XN:YO=YN
130 XN=XN+DX:IFXN=0ORXN=MXTHENDX=-DX
135 YN=YN+DY:IFYN=0ORYN=MYTHENDY=-DY
140 X=XN:Y=YN:GOSUB1300:FORT=0TO10:NEXT
145 X=XN:Y=YN:GOSUB1350
150 NEXT
1250 REM PUNTO
1255 P1=PEEK(BV+A):H=16
1260 FORK=0TO15:IFP1=D(K)THENH=K:K=15
1265 NEXT:RETURN
1300 REM DISEGNA
1305 IFX<0ORY<0ORINT(X)>79ORINT(Y)>49THEN1390
1310 A=40*(24-INT(Y/2))+INT(X/2)
1315 GOSUB1250:IFH=16THENH=0:IFNPTHENRETURN
1320 CH=D(H OR (2↑(X AND 1)*4↑(Y AND 1)))
1325 POKEBV+A,CH:POKEBC+A,CO
1330 RETURN
1350 REM CANCELLA
1355 IFX<0ORY<0ORINT(X)>79ORINT(Y)>49THEN1390
1360 A=40*(24-INT(Y/2))+INT(X/2)
1365 GOSUB1250:IFH=16THENH=0:IFNPTHENRETURN
1370 CH=D(HAND(15-(2↑(X AND 1)*4↑(Y AND 1))))
1375 POKEBV+A,CH:POKEBC+A,CO
1380 RETURN
1390 PRINT"?FUORI DAL VIDEO  ERRORE":END
```

1.5.3 Istogrammi

Per ottenere buoni istogrammi possiamo usare caratteri grafici appartenenti ai 4 gruppi riportati nella Figura 1.8.

CAR.	TASTI	D-CODE	CAR.	TASTI	D-CODE
	CBM+T	99		CBM+@	99
	CBM+Y	119		CBM+P	119
	CBM+U	120		CBM+O	120
	RVS/CBM+I	226		CBM+I	226
	RVS/CBM+O	249		RVS/CBM+U	249
	RVS/CBM+P	239		RVS/CBM+Y	239
	RVS/CBM+@	228		RVS/CBM+T	228
	CBM+G	101		CBM+M	101
	CBM+H	116		CBM+N	116
	CBM+J	117		CBM+L	117
	CBM+K	97		RVS/CBM+K	97
	RVS/CBM+L	246		RVS/CBM+J	246
	RVS/CBM+N	234		RVS/CBM+H	234
	RVS/CBM+M	231		RVS/CBM+G	231

Figura 1.8 Caratteri grafici usati per tracciare ISTOGRAMMI

In questi gruppi il carattere è considerato come composto da 8 barrette, elementi orizzontali o verticali di 8 punti ciascuno. Utilizzando questi caratteri grafici insieme allo spazio e allo spazio inverso (rettangolo vuoto o rettangolo pieno), possiamo creare sullo schermo dei grafici a barre (istogrammi). La risoluzione orizzontale è di 40*8=320 barrette e quella verticale di 25*8=200 barrette.

Per comprendere il metodo usato per creare una barra immagina di aver calcolato che essa debba essere lunga 263 elementi. Per disegnarla è sufficiente scrivere INT(263/8) caratteri spazio inverso, e, supponendo che la barra debba essere disegnata da sinistra a destra, il carattere CBM-M in campo inverso, che contiene 263-INT(263/8)*8=263-32*8=263-256=7 barrette.

Per fare comparire la barra sul video si possono usare almeno due procedimenti diversi:

- per intero, cioè stampare gli spazi in campo inverso e, eventualmente, il necessario carattere di completamento;

• gradualmente, cioè stampare uno sull'altro di seguito i caratteri appartenenti allo stesso gruppo per dare l'impressione di una barra che cresce.
In entrambi i casi possono essere usati sia PRINT che POKE; come vediamo nel programma ES35.

```
1 REM ES35
3 REM ISTOGRAMMI
5 DIM A$(7):RESTORE
10 FORK=0TO7:READA,B:A$(K)=CHR$(B)
13 IFATHENA$(K)="▣"+A$(K)+"■"
15 NEXTK
20 PRINT"▢▣";:FORK=1TO10:READA
25 PRINT"BARRA DA"A":":N=A:GOSUB3000:PRINT
30 NEXTK:END
35 END
40 DATA0,32,0,180,0,165,0,181,0,161,1
41 DATA182,1,170,1,167
42 REM DATI PER BARRE
44 DATA 1,2,7,18,57,101,175,206,320,295
3000 REM BARRE SX TO DX
3005 N1=INT(N/8)
3010 IFN1THENPRINT"▣";
3013 FORI=1TON1:PRINTCHR$(32);:NEXT:PRINT"■";
3015 N1=N-N1*8
3020 PRINTA$(N1);:RETURN
```

Per vedere crescere più lentamente le barre puoi provare il programma ES35.1 che segue.

```
1 REM ES35.1
3 REM ISTOGRAMMI
5 DIM A$(7):RESTORE
10 FORK=0TO7:READA,B:A$(K)=CHR$(B)
13 IFATHENA$(K)="▣"+A$(K)+"■"
15 NEXTK
20 PRINT"▢▣";:FORK=1TO10:READA
25 PRINT"BARRA DA"A":":N=A:GOSUB3000:PRINT
30 NEXTK:END
35 REM DATI PER A$()
40 DATA0,32,0,180,0,165,0,181,0,161,1
```

```
41 DATA182,1,170,1,167
42 REM DATI PER BARRE
44 DATA 1,2,7,18,57,101,175,206,320,295
3000 REM BARRE SX TO DX
3005 N1=INT(N/8)
3010 IF N1=0THEN3025
3015 FORI=1TON1:FORJ=0TO7:PRINTA$(J)"▌";
3017 FORT=1TO20:NEXT
3020 NEXT:PRINT"▘ ▀";:NEXT
3025 N1=N-N1*8
3030 FORJ=0TON1:PRINTA$(J)"▌";
3033 FORT=1TO20:NEXT
3035 NEXT:PRINT"▌";:RETURN
```

Per creare istogrammi verticali le routine sono quasi identiche: in quelle con la PRINT bisogna posizionare il carattere successivo di sopra e non di fianco all'ultimo carattere disegnato. Inoltre se fai iniziare l'istogramma in basso a destra provocherai lo scrolling. Usando le POKE, invece, cambia solo il modo di variare PV. La routine viene modificata come indicato in ES36.

```
1 REM ES36
5 REM ISTOGRAMMI VERTICALI
10 FORK=0TO7:READA,B:A$(K)=CHR$(B)
15 IFATHENA$(K)="▘"+A$(K)+"▀"
20 NEXT
25 PRINT"▛";:FORK=1TO8:READA
30 PRINT"▚"SPC(255)SPC(255)SPC(255)SPC(195);
35 PRINTTAB(1+4*K+(A>9)+(A>99))A"▌▌▌▌";
40 PRINTTAB(1+4*K)"▜";:N=A:GOSUB3000:NEXT
43 GOTO43
44 REM DATI PER A$()
45 DATA0,32,0,164,0,175,0,185,0,162,1
46 DATA184,1,183,1,163
47 REM
48 REM DATI PER IL GRAFICO
49 DATA100,27,180,39,49,87,131,90
3000 REM ROUTINE BARRE BOT TO TOP VELOCE
3005 N1=INT(N/8)
3010 IFN1=0THEN3020
```

```
3015 PRINT"█";:FORI=1TON1:PRINT" █▯";:NEXT
3020 PRINT"■";:N1=N-N1*8
3025 PRINTA$(N1);:RETURN
```

Sulla cassetta sono registrati anche i programmi ES36.1 e ES36.2; essi sono una variazione di ES36. Il primo fa crescere le barre più lentamente, il secondo le traccia colorate.

Non includiamo anche tutte le altre routine possibili, in quanto occuperebbero molte pagine, e, inoltre, esse sono molto semplici da ricavare basandosi sugli esempi esposti.
Come al solito abbiamo una limitazione nel numero dei colori; ogni gruppo di 8 barrette può contenere elementi di un colore a scelta tra i 16 possibili o del colore dello sfondo. A volte può essere utile disegnare grafici con barre al di sopra e al di sotto di una linea orizzontale di demarcazione, o a sinistra e a destra di una linea verticale. Per ottenerli bisogna avere l'accortezza di fare coincidere le origini delle coppie di barre e usare due colori diversi.

Segue un esempio di questo metodo nel programma ES37.

```
1 REM ES37
5 REM ISTOGRAMMI A 2 COLORI
10 PRINT"▀";
20 DIMSU$(7),GIU$(7)
30 FORK=0TO7:READSU$(K),GIU$(K):NEXT
40 FORK=1TO30:READ V:CO=-2*(V>0)
50 PRINT"▒"SPC(255)SPC(185)SPC(4+K-40*(V<0));
55 GOSUB100
60 NEXT
70 PRINT"▒";:POKE646,6
80 GOTO80
100 REM ISTO. SU E GIU
105 CH=INT(ABS(V)/8):P=ABS(V)-CH*8
106 POKE646,CO
110 IFV<0THEN130
115 REM ISTOGRAMMA IN SU
120 IF CH THEN FORJ=1TOCH:PRINT"█ ■▯";:NEXT
125 PRINTSU$(P);:RETURN
130 REM ISTOGRAMMA IN GIU'
```

```
135 IF CH THEN FORJ=1TOCH:PRINT"  ";:NEXT
140 PRINTGIU$(P);:RETURN
8000 DATA" "," ","_","‾","▄","‾","▄"
8003 DATA"‾","▄",""
8005 DATA"","","","","",""
8010 DATA1,2,3,10,20,-6,-4,-1,-50,+80
8015 DATA0,11,-10,-30,-55,-23,11,90,-96,3
8020 DATA49,47,-50,-25,77,80,-80,-5,-1,1
```

1.6 ESEMPI DI USO DEI CARATTERI GRAFICI

Seguono alcuni programmi o sottoprogrammi esempio che sfruttano i caratteri grafici disponibili.

QUASI ANIMAZIONE IN BASIC: usando un sistema simile a quello usato per gli istogrammi, disegnamo un quadratino che si muove sul video lungo una linea. Usiamo i caratteri riportati in Figura 1.9.

Figura 1.9 Coppie di caratteri da usare per produrre animazione

Il programma ES38 che segue fa muovere un quadratino sulla quarta linea dello schermo; come vedi il programma è abbastanza semplice.

```
5 REM ES38
10 DIMA(7)
12 S$="
15 FORK=0TO7:READA(K):NEXT
20 DATA101,116,117,97,246,234,231,160
30 PRINT"";S$;"";
```

```
35 BA=1144:H=10:C=0
40 FORK=0TO1:K=0
45 A=128+256*(A(C)>127)
50 POKE BA+H,A(C)+A
55 POKE BA+H+1+(1+H)*(H=39),A(C)
60 C=C+1+8*(C=7)
63 IFC=0THENPOKEBA+H,32:H=H+1+40*(H=39)
65 NEXT
```

Il calcolo di A permette di evitare l'uso di due vettori o di un vettore più lungo, dato che i due caratteri da stampare di seguito sono uno l'inverso dell'altro. La PRINT alla linea 20 pone nella mappa colore il codice colore del cursore in tutta la terza linea dello schermo. Un ciclo avrebbe occupato meno memoria. Il ciclo da 40 a 65 non ha mai termine.

TERMOMETRO: usiamo questa volta i 16 caratteri grafici che mostrano ognuno una barretta singola in ciascuna delle 8 possibili posizioni; essi sono riportati nella Figura 1.10.

CAR.	TASTI	D-CODE	CAR.	TASTI	D-CODE
|	CBM+G	101	‾	CBM+T	101
|	SHIFT+T	84	‾	SHIFT+E	84
|	SHIFT+G	71	‾	SHIFT+D	71
|	SHIFT+B	66	-	SHIFT+C	66
|	SHIFT+-	93	-	SHIFT+*	93
|	SHIFT+H	72	_	SHIFT+F	72
|	SHIFT+Y	89	_	SHIFT+R	89
|	CBM+M	103	_	CBM+@	103

Figura 1.10 Caratteri a barretta singola

Nel programma ES39 usiamo le barrette verticali per simulare il movimento dell'aghetto di un termometro che riporta i dati termici relativi ad un ipotetico paziente (i dati sono prelevati da un DATA, ma l'ingresso potrebbe essere ottenuto da tastiera o anche collegando il calcolatore a un termometro reale).

```
1 REM ES39
5 REM TERMOMETRO
6 DIMA$(7):RESTORE:C$="▒"
7 FORK=0TO7:READA:A$(K)=CHR$(A):NEXT
8 DATA165,212,199,194,221,200,217,167
9 CN$="3 6  3 7  3 8  3 9  4 0  4 1  4 2  4 3"
10 DATA37,39,35,38.5,40.5,44,37.75,-1
13 S$="":FORK=1TO38:S$=S$+" ":NEXTK
15 PRINT"▢  "+CN$;
20 FORK=1TO8:PRINT"───┼─";:NEXT
25 PRINT"▒"S$"▒";
30 FORK=1TO8:PRINT"───┴─";:NEXT
35 BS=35.5:FS=43.05:RI=40:XP=0
40 RESTORE:FORK=1TO8:READA:NEXT
45 FORK=0TO1:K=0
50 READDA:IFDA=-1THENK=1:NEXT:GOTO40
55 X=(DA-BS)*RI:GOSUB100:XP=X:NEXT
60 END
100 REM SPOSTA AGO
105 IFX=XPTHENRETURN
110 IFXP<0THENXP=0
115 IFXP>303THENXP=303
120 FORJ=XPTOXSTEPSGN(X-XP):PRINT"▓▓▓";
125 IFJ>=0ANDJ<304THEN145
130 IFJ<0THENPRINT"▓"C$"▢"S$C$;:GOTO140
135 IFJ>303THENPRINTC$S$"▓"C$"▢";
140 J=X:FORL=1TO150:NEXT:NEXT:RETURN
145 PO=INT(J/8):CH=J-PO*8
150 PRINTC$LEFT$(S$,PO)A$(CH)LEFT$(S$,37-PO)C$;
155 NEXT:FORL=1TO150:NEXTL:RETURN
```

Il valore minimo rappresentabile è 35.5, il massimo 43.5. Il numero di posizioni possibili per l'ago è 38*8 (poichè il primo e l'ultimo carattere della linea sono occupati), cioè 304.
La differenza di temperatura tra una posizione dell'ago e la successiva è 0.025.

SCACCHIERE: ne puoi ottenere di tutte le dimensioni. Ti mostreremo tre esempi, ma ne potrai creare tanti altri. In Figura 1.11 sono riportati 3 tipi di scacchiera e seguono i 3 programmi per ottenerle: ES41.1, ES41.2 e ES41.3.

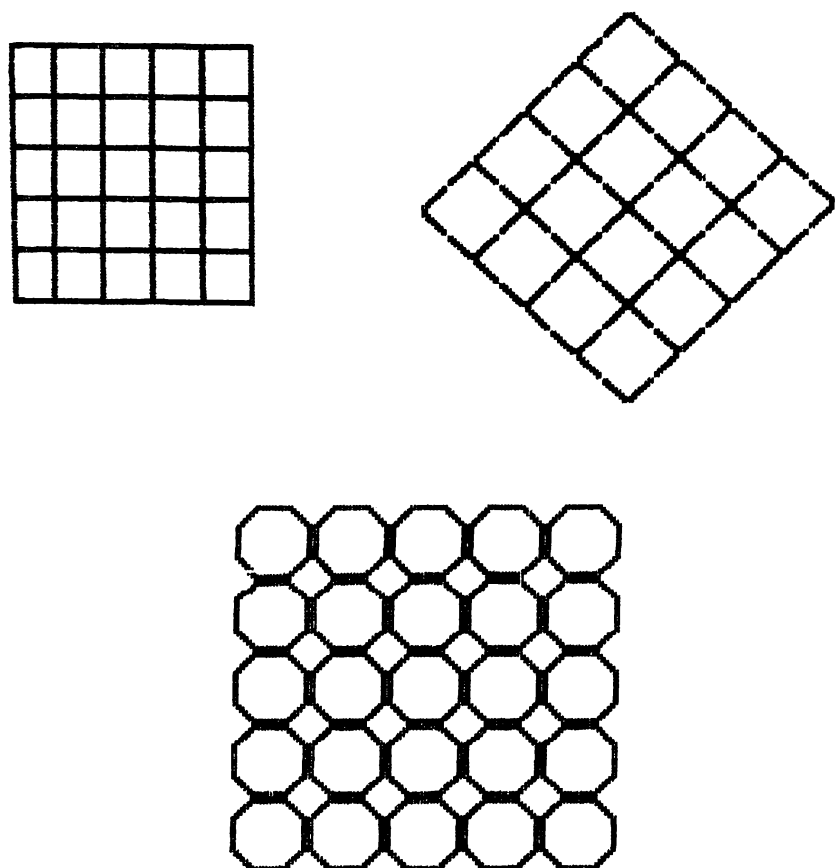



Figura 1.11 Scacchiere ottenute con i programmi ES41.1, ES41.2 E ES41.3

```
1 REM ES41.1
5 REM SCACCHIERA TIPO 1
10 INPUT"NUMERO QUADRATI/LINEA (MAX 12)";NL
13 NL=INT(NL)
15 NL=NL-NL*(NL<0)+(12-NL)*(NL>12):PRINT"";
20 PRINT" r";
```

```
23 IFNL>1THENFORK=1TONL-1:PRINT"─┬";:NEXT
25 PRINT"─┐"
30 IFNL=1THEN60
40 FORJ=1TONL-1
45 FORK=1TONL:PRINT"│ ";:NEXT:PRINT"│"
50 PRINT"├";:FORK=1TONL-1:PRINT"─┼";:NEXT
53 PRINT"─┤"
55 NEXT
60 FORK=1TONL:PRINT"│ ";:NEXT:PRINT"│"
65 PRINT"└";
67 IFNL>1THENFORK=1TONL-1:PRINT"─┴";:NEXT
70 PRINT"─┘";
75 GOTO75
```

```
 1 REM ES41.2
 5 REM SCACCHIERA TIPO 2
 10 PRINT"[illegible]";
 20 FORK=1TO4
 25 FORJ=1TOK
 30 PRINTTAB(20-2*K+4*(J-1))"╱╲";
 35 NEXT:PRINT
 40 FORJ=1TOK
 45 PRINTTAB(19-2*K+2*(J-1))"╱  ╲";
 50 NEXT:PRINT:NEXT
 60 FORK=4TO1STEP-1
 65 FORJ=1TOK
 70 PRINTTAB(19-2*K+2*(J-1))"╲  ╱";
 75 NEXT:PRINT
 80 FORJ=1TOK
 85 PRINTTAB(20-2*K+4*(J-1))"╲╱";
 90 NEXT:PRINT:NEXT
 95 GOTO95
```

```
1 REM ES41.3
5 REM SCACCHIERA TIPO 3
10 INPUT"[illegible]RIGHE (MAX 8)";R%
12 IFR%<1ORR%>8THEN10
15 INPUT"[illegible]COLONNE (MAX 13)";C%
```

```
17 IFC%<1ORC%>13THEN15
20 PRINT"□";
25 FORJ=1TOR%
30 FORK=1TOC%:PRINT"╱‾╲";:NEXT:PRINT
40 FORK=1TOC%:PRINT"|  |";:NEXT:PRINT
50 FORK=1TOC%:PRINT"╲_╱";:NEXT:PRINT
55 NEXT
60 GOTO60
```

TASTIERA: Il programma ES42 disegna una tastiera con 19 tasti bianchi e 14 tasti neri, coprendo due ottave. Può essere uno spunto se sei appassionato di musica, per creare anche programmi didattici, ad esempio per visualizzare gli accordi.

```
1 REM ES42
705 REM TASTIERA
710 A$=" | ■ ■ ■ ■ | ■ ■ ■ ■ ■ ■ "
715 PRINT"□■";
720 PRINT" ■";:FORK=1TO38:PRINT"■";
723 NEXT:PRINT
725 FORK=1TO5:PRINTA$A$LEFT$(A$,19):NEXT
730 FORJ=1TO4
735 PRINT" |";:FORK=1TO19:PRINT"  |";:NEXT
740 PRINT:NEXT
745 PRINT" └";:FORK=1TO18:PRINT"─┴";
746 NEXT:PRINT"─┘"
750 GOTO750
```

ITALIA: abbiamo preparato tre esempi, i primi due, ES43 e ES44, in bassissima risoluzione, il secondo, ES45, in bassa risoluzione. Potranno servirti di spunto per disegnare sullo schermo.

IN BASSISSIMA RISOLUZIONE: si possono scegliere diverse strade:

- usare una serie di PRINT con le stringhe alfanumeriche già preparate,
- usare una serie di PRINT che leggano da un file interno, DATA, le stringhe da stampare,
- codificare in qualche modo il disegno.

Noi abbiamo scelto l'ultima strada poichè il nostro disegno richiede solo due differenti caratteri. Stabilito ciò, abbiamo analizzato il disegno che volevamo

ottenere e abbiamo definito una serie di stringhe numeriche, ciascuna delle quali descrive una linea in questo modo:

XX1 YY1 XX2 YY2...

dove:

XXi è il numero di caratteri di tipo A,
YYi è il numero di caratteri di tipo B.

Noi abbiamo usato come carattere A lo spazio e come carattere B lo spazio inverso, ma si può usare una coppia diversa di caratteri, apportando le necessarie modifiche.

```
1 REM ES43
5 REM ITALIA
10 RESTORE:READNL:PRINT"■";
20 FORC=1TONL:READA$
30 FORI=1TOLEN(A$)STEP4
40 A=VAL(MID$(A$,I,2)):PRINTSPC(A);
50 A=VAL(MID$(A$,I+2,2))
55 FORK=1TOA:PRINT"■ ■";:NEXT
60 NEXT:IF POS(0)<39AND C<NL THENPRINT
70 NEXT
80 PRINT"■"SPC(180)"ITALIA";
90 GOTO90
1000 DATA25,0903,0706,03010109,030101080101
1005 DATA0110,0210,0111,0112,03020307
1010 DATA0808,0909,08010112,1209,1312
1015 DATA06011010,0602090603 01,05031203
1020 DATA06021403,05031502,05031501,06011601
1025 DATA21010101,1606,1605,1903
```

Una modifica può essere usare dei caratteri "O" invece degli spazi e attribuire i giusti colori a terra e mare, come segue:

```
1 REM ES44
5 REM ITALIA
10 RESTORE:READNL:PRINT"■";
13 POKE53280,1:POKE53281,1
15 PRINT"■";
20 FORC=1TONL:READA$
```

```
30 FORI=1TOLEN(A$)STEP4
40 A=VAL(MID$(A$,I,2))
43 FORK=1TOA:PRINT"●";:NEXT
45 PRINT"■";
50 A=VAL(MID$(A$,I+2,2))
53 FORK=1TOA:PRINT"▪ ■";:NEXT
55 PRINT"▪";
60 NEXT:IFPOS(0)<39THEN100
65 IF C<NLTHENPRINT
70 NEXT
80 PRINT"▪"SPC(180)"▪ITALIA";
90 GOTO90
100 FORK=POS(0)TO28:PRINT"●";:NEXT:GOTO65
1000 DATA25,0903,0706,03010109,030101080101
1005 DATA0110,0210,0111,0112,03020307
1010 DATA0808,0909,08010112,1209,1312
1015 DATA06011010,060209060301,05031203
1020 DATA06021403,05031502,05031501
1025 DATA06011601,21010101,1606,1605,1903
```

IN BASSA RISOLUZIONE: questa volta abbiamo usato un diverso sistema di codifica, associando ai 16 possibili caratteri della bassa risoluzione altrettante lettere dell'alfabeto, come risulta dalla Figura 1.12.

```
CORRISPONDENZA LETTERE/CARATTERI

A SPAZIO   B  ▖      C  ▗      D  ▄

E  ▘       F  ▌      G  ▚      H  ▙

I  ▝       J  ▞      K  ▐      L  ▟

M  ▀       N  ▛      O  ▜      P  █
```

Figura 1.12 Esempio di codifica di caratteri per GRAFICA a BASSA RISOLUZIONE

Ogni linea è stata codificata come:

XX,YYYYYY,XX,YYY,...,-1

dove:

XX indica il numero degli spazi,
YY..YY indica le lettere da usare e quindi i caratteri grafici,
il numero -1 chiude la linea.

Per diminuire la possibilità di errori abbiamo usato un'istruzione DATA per ogni linea da disegnare.

```
 1 REM ES45
 3 REM ITALIA
 5 RESTORE:DIMA$(15)
 6 FORK=0TO15:READA$(K):NEXT
 10 READNL:PRINT"▚▉▉";
 15 POKE53280,0:POKE53281,0:POKE646,7
 20 FORK=1TONL:PRINTSPC(5);
 30 FORJ=0TO1:J=0:READA
 40 IFA=-1THENJ=1:NEXT:GOTO90
 50 PRINTSPC(A);
 60 READA$
 70 FORI=1TOLEN(A$):A=ASC(MID$(A$,I,1))-65
 75 PRINTA$(A);:NEXT
 80 NEXT
 90 PRINT:NEXT:PRINT"▚"SPC(180)"ITALIA";
 95 GOTO95
100 DATA" ","▖ "," ▗","▄▄","▀ ","▌ ","▝▖","▚▀"
105 DATA"▀","▞▄","▚▀","▞▀","▙▄"
110 DATA"▚▄","▙▀","▚▀"
1000 REM DATA PER <<ITALIA>>
1001 DATA22,6,CL,-1
1002 DATA5,LLPHD,-1
1003 DATA2,HKLLPPPH,-1
1004 DATA0,KPPLPPPPPMF,-1
1005 DATA0,LPPPPPPP,-1
1006 DATA0,OPPPPPPPE,-1
1007 DATA0,OPNMPPPPH,-1
1008 DATA1,J,2,EPPPPD,-1
1009 DATA5,KPPPPF,-1
1010 DATA5,JPPPH,-1
```

```
1011 DATA6,IOPPPH,-1
1012 DATA7,IPPPPPPE,-1
1013 DATA8,IOPPPPHD,-1
1014 DATA2,BLB,6,OPPPPPD,-1
1015 DATA2,OPF,6,IMPPNIOF,-1
1016 DATA2,KPF,8,IPF,2,E,-1
1017 DATA2,LPF,9,KPB,-1
1018 DATA2,ONE,10,NE,-1
1019 DATA14,KF,-1
1020 DATA9,CDDDPI,-1
1021 DATA9,OPPPF,-1
1022 DATA10,IMPP,-1
```

MARYLIN: ecco, in ES46, le linee DATA per realizzare su 25 linee un ritratto, un po' allungato, di MARYLIN MONROE.

```
 1 REM ES46
 3 REM MARYLIN
 5 RESTORE:DIMA$(15)
 6 FORK=0TO15:READA$(K):NEXT
 10 READNL:PRINT"♥";
 15 POKE53280,0:POKE53281,0:POKE646,8
 20 FORK=1TONL:PRINTTAB(3);
 30 FORJ=0TO1:J=0:READA
 40 IFA=-1THENJ=1:NEXT:GOTO90
 50 PRINTSPC(A);
 60 READA$
 70 FORI=1TOLEN(A$):A=ASC(MID$(A$,I,1))-65
 75 PRINTA$(A);:NEXT
 80 NEXT
 90 IFPOS(0)<39ANDK<NLTHENPRINT
 95 NEXT:PRINT"▚"SPC(31)"▀MARYLIN";
 98 GOTO98
100 DATA" ","▗ ","▖ ","▄ ","▘ ","▌ ","▚ ","▖▀"
105 DATA"▀","▞▟","▖▀","▞▀","▙▟"
106 DATA"▖▟","▙▀","▖▀"
999 REM DATA PER <<MARYLIN>>
1000 DATA25,1,PPPPEADPPPPPPPPPPPPPPPPPPPH,-1
1001 DATA0,KPPPPALPPPPPPPPPPPPPPPPPPPPB,-1
```

```
1002 DATA0,KPPPPKPPPPPPPPPPPPNMPPPPPPPH,-1
1003 DATA1,PNBPPPPPPPPPPPPEDDDOPPPPOPPB,-1
1004 DATA1,IPHIPPPPPPPPPNDPPPPPPPPPEKPPH,-1
1005 DATA1,EPPPOPPOPPPPPPPPPPPPPPPPAAIOP,-1
1006 DATA1,IIPPKPNEKPPPPPPNMMNEPPPPADBAO,-1
1007 DATA2,EOPAEDNMMPPPNDDDLPPPPLPPHAPB,-1
1008 DATA2,CJPACPPPHOPPPPPPPPPPPPPPPPBPF,-1
1009 DATA2,IAPAKPNELKPPPPPPPPPPPPPPPPPP,-1
1010 DATA3,LFAIACLPPPPPPPPPPPPPPPPPPPPP,-1
1011 DATA3,IHBACPPPKPPPPPPPPPPPPPPPPPPO,-1
1012 DATA3,KPPAKPPFIPNDLPPHOPPPKPPPN,-1
1013 DATA3,LNPAPPPHLPPPPPPPPHOPPKPPPECF,-1
1014 DATA3,OPIAOPPPPPPPNNMPPPPPPNOPNALF,-1
1015 DATA3,IPPAAOPPONACDDBPPPPPPPFAAAPP,-1
1016 DATA4,IPBAAOPKALPMCBPPPPPPPFACPPPB,-1
1017 DATA3,CAPPACKPNAMCLNCPPPPPPPFAAALPF,-1
1018 DATA3,KKPPBABPPBCPEBLPPPPPPPFADCPPF,-1
1019 DATA3,BAIOHACIPHAAILPPPPHPPFAPPPNE,-1
1020 DATA3,MAAAOBAHOPHDPPPPPPNLPPFAKPNHDDD,-1
1021 DATA4,FAAAAAIFOPPPPPPPPKPPPFKPNLPPPPPD,-1
1022 DATA4,IAAAAAAICPPPPPPPELPPPFLNCPPPPPPP,-1
1023 DATA5,GAAALDPHJOPNEALPPPPFMCPPPPPPPP,-1
1024 DATA6,GALPPPPPPHAAKPPPPPFALPPPPPPPP,-1
```

CARTE DA GIOCO: dato che tra i caratteri grafici sono compresi i 4 semi delle carte francesi, abbiamo preparato una routine per disegnare le carte, ES47.

```
1 REM ES47
5 SE$="♥♦♣♠":N$="*A234567891JQK"
8 POKE53281,1
10 C$="■":C1$=" |        | ":A$=" |   ":B$="   | "
15 CC=646:POKECC,6
18 SP$="▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶"
20 INPUT"■CARTA (0=JOLLY 13=KING=RE)";V
25 IFV<0ORV>13THEN20
30 INPUT"■■■SEME (1=♥,2=♦,3=♣,4=♠)";S
35 IFS<1ORS>4THEN30
40 GOSUB3000:GOSUB3030:GOSUB3050
45 POKE198,0
```

```
50 GETA$:IFA$=""THEN50
55 RUN
3000 PRINT"┌──────────┐"SPC(30);
3005 FORK=0TO11:PRINT"|▒▒▒▒▒▒▒▒▒▒|"SPC(30);
3010 NEXT
3015 PRINT"└──────────┘";
3020 PRINT"TTTTTTTTTTTT██████████";
3025 RETURN
3030 PRINT"█";
3035 FORK=0TO11:PRINT"|          |"SPC(30);:NEXT
3038 PRINT"TTTTTTTTTTTTT█";
3040 RETURN
3050 D$=MID$(SE$,S,1):IFV=0THEND$=" "
3052 PRINTMID$(C$,2+(S<3),1);
3053 PRINTMID$(N$,V+1,1);:IFV=10THENPRINT"0█";
3054 PRINT"█└─────"SP$D$;
3055 V=V+1:IFV<1ORV>14THENRETURN
3060 IFV>7THENV=V-7:GOTO3075
3065 ONVGOSUB3350,3100,3115,3140,3160,3180,3200
3070 GOTO3080
3075 ONVGOSUB3220,3250,3270,3290,3320,3330,3340
3080 PRINT"█┌     TTTTTTTTTTTT█████";:RETURN
3100 FORK=1TO4:PRINTC1$SP$;:NEXT
3105 PRINT"|  "D$"  |"SP$;
3110 FORK=1TO4:PRINTC1$SP$;:NEXT:RETURN
3115 PRINTC1$SP$C1$SP$A$D$B$;
3125 PRINTSP$C1$SP$C1$SP$C1$;
3130 PRINTSP$C1$SP$A$D$B$;
3135 PRINTSP$C1$SP$;:RETURN
3140 PRINTC1$SP$;
3145 FORK=1TO3:PRINTC1$SP$A$D$B$;
3150 PRINTSP$;:NEXT
3155 PRINTC1$SP$C1$SP$;:RETURN
3160 PRINTC1$SP$C1$SP$"| "D$" "D$" |";
3165 PRINTSP$C1$SP$C1$SP$C1$;
3170 PRINTSP$"| "D$" "D$" |"SP$C1$;
3175 PRINTSP$C1$SP$;:RETURN
3180 PRINTC1$SP$C1$SP$"| "D$" "D$" |";
3185 PRINTSP$C1$SP$A$D$B$SP$C1$;
3190 PRINTSP$"| "D$" "D$" |"SP$C1$;
3195 PRINTSP$C1$SP$;:RETURN
3200 PRINTC1$SP$C1$SP$;
```

```
3205 FORK=1TO3:PRINT" | "D$" "D$" |"SP$;
3210 PRINTC1$SP$;:NEXT:PRINTC1$SP$;
3215 RETURN
3220 PRINTC1$SP$C1$SP$;
3225 PRINT" | "D$" "D$" |"SP$;
3230 PRINTA$D$B$SP$;
3235 FORK=1TO2:PRINT" | "D$" "D$" |"SP$;
3240 PRINTC1$SP$;:NEXT
3245 PRINTC1$SP$;:RETURN
3250 PRINTC1$SP$;
3255 FORK=1TO4:PRINT" | "D$" "D$" |"SP$;
3260 PRINTC1$SP$;:NEXT:RETURN
3265 RETURN
3270 PRINTC1$SP$" | "D$" "D$" |"SP$;
3275 PRINTA$D$B$SP$;
3280 FORK=1TO3:PRINT" | "D$" "D$" |"SP$C1$;
3285 PRINTSP$;:NEXT:RETURN
3290 PRINTC1$SP$;
3295 FORK=1TO2:PRINT" | "D$" "D$" |";
3300 PRINTSP$A$D$B$SP$;:NEXT
3305 FORK=1TO2:PRINT" | "D$" "D$" |"SP$C1$SP$;
3310 NEXT:RETURN
3320 PRINT" | ◥▒ ▄◤ ⅃ "SP$;
3321 PRINT" | ▝▒▔▀ || "SP$;
3322 PRINT" | ▒◤"D$"▄◤▒|▄ "SP$;
3323 PRINT" |▒◤"D$"/─┘▄ "SP$;
3324 PRINT" |▒"D$"─┼─"D$"▄ "SP$;
3325 PRINT" |▒┌─/"D$"▄◤┐ "SP$;
3326 PRINT" |▒|◤"D$"▄◤ | "SP$;
3327 PRINT" || ▪▒▙▄▙ | "SP$;
3328 PRINT" ⊢▒◤ ◥▄ | "SP$;
3329 RETURN
3330 PRINT" | ◥▒ ▄◤ ⅃ "SP$;
3331 PRINT" | ▒◤O◥▄H "SP$;
3332 PRINT" |▒◤▔ ▄◤▒|▄ "SP$;
3333 PRINT" |▒╲"D$"╲╲╱▄ "SP$;
3334 PRINT" |▒╲╲"D$"╲╲▄ "SP$;
3335 PRINT" |▒╱╲╲"D$"╲▄ "SP$;
3336 PRINT" |▒|◤ ▁▄◤┐ "SP$;
3337 PRINT" H◥▒▫▄◤ | "SP$;
```

```
3338 PRINT" ⊢▛ ▜ | "SP$;
3339 RETURN
3340 PRINT" | ▜▟ ▝▟|| "SP$;
3341 PRINT" | ▝▟▛▀ ⊣| "SP$;
3342 PRINT" |▛ "D$"▀▛▟|█ "SP$;
3343 PRINT" |▟ "D$"/—┘█ "SP$;
3344 PRINT" |▟"D$"/+/"D$"█ "SP$;
3345 PRINT" |▟┌─/"D$" █ "SP$;
3346 PRINT" |▟|▛"D$" ▀┐ "SP$;
3347 PRINT" ⊢▪▟▙▙ | "SP$;
3348 PRINT" ||▟▖ ▀▖ | "SP$;
3349 RETURN
3350 PRINTC1$SP$" | ****| "SP$;
3351 PRINT" |  ** | "SP$;
3352 PRINT" |  ** | "SP$;
3353 PRINT" |* ** | "SP$;
3354 PRINT" |* ** | "SP$;
3355 PRINT" |* ** | "SP$;
3356 PRINT" | **  | "SP$C1$SP$;:RETURN
```

CIFRE GRANDI: seguono due routine per ingrandire le cifre 2*3 e 5*4.

Ingrandimento 2*3 con il programma ES48 : vogliamo generare cifre come quelle riportate nella Figura 1.13, per scrivere un numero intero N.

Figura 1.13 Ingrandimenti cifre 2X3

Come per le carte da gioco possiamo usare 10 piccole routine che disegnano ognuna delle 10 cifre.

```
1 REM ES48
5 REM CIFRE 2X3
10 INPUT"NUMERO:  ";N
```

```
20 GOSUB200
99 GOTO99
200 N$=MID$(STR$(N),2,LEN(STR$(N))-1)
205 G$=""
210 FORK=1TOLEN(N$):A=VAL(MID$(N$,K,1))+1
220 ONAGOSUB250,252,254,256,258
221 IFA<=5THEN225
223 ON(A-5)GOSUB260,262,264,266,268
225 PRINT"";:NEXT:RETURN
250 PRINT"  "G$"  "G$"  ";:RETURN
252 PRINT"  "G$"  "G$"  ";:RETURN
254 PRINT"  "G$"  "G$"  ";:RETURN
256 PRINT"  "G$"  "G$"  ";:RETURN
258 PRINT"  "G$"  "G$"  ";:RETURN
260 PRINT"  "G$"  "G$"  ";:RETURN
262 PRINT"  "G$"  "G$"  ";:RETURN
264 PRINT"  "G$"  "G$"  ";:RETURN
266 PRINT"  "G$"  "G$"  ";:RETURN
268 PRINT"  "G$"  "G$"  ";:RETURN
```

Ingrandimento 5*4 con il programma ES49: possiamo ingrandire ancora le cifre precedenti, usando un sistema di codifica procediamo così: abbiamo osservato che per ogni riga le possibili combinazioni di punti sono quelle riportate nella Figura 1.14.

Figura 1.14 Combinazioni di punti per ingrandire cifre 5X4

e che ogni cifra può essere descritta con 5 numeri che corrispondono alle combinazioni di punti da utilizzare. Nella Figura 1.15 abbiamo rappresentato tutte le cifre ingrandite.

Figura 1.15 Ingrandimento cifre 5X4

```
1 REM ES49
5 REM CIFRE 5X4
10 DIMA$(6):DIMB$(9)
20 FORK=0TO6:READA$(K):NEXTK
30 FORK=0TO9:READB$(K):NEXTK
33 DATA"   ▞ ▀","  ▞ ▀ ","  ▞  ▀","▞ ▀  "
34 DATA"▞ ▀ ▞ ▀","▞   ▀","▞  ▀ "
35 DATA65556,27226,61646,61316,44561,64616
36 DATA64656,61111,65656,65611
50 INPUT"NUMERO: ";N
60 GOSUB200
99 GOTO99
200 N$=MID$(STR$(N),2,LEN(STR$(N))-1)
205 G$="▮▮▮▮"
210 FORK=1TOLEN(N$):A=VAL(MID$(N$,K,1))
220 FORI=1TO5:B=VAL(MID$(B$(A),I,1))-1
230 PRINTA$(B)"▮▮▮▮▮";
240 NEXTI
250 PRINT"IIIII▮▮▮▮";
260 NEXTK
270 RETURN
```

LETTERE E NUMERI: usando i caratteri grafici possiamo ingrandire lettere e numeri. Nei due esempi precedenti abbiamo usato due metodi diversi di ingrandimento; un metodo può essere quello di leggere in ROM le descrizioni dei caratteri e

poi sostituire a ogni puntino un carattere grafico adatto (sia in bassissima che in bassa risoluzione). Altro metodo può essere quello di preparare per ogni carattere una stringa che lo descrive, oppure, fissate le dimensioni volute, preparare delle stringhe che contengano ognuna una parte di tutti i caratteri, nelle quali si possono ritrovare le sottostringhe che compongono il carattere ingrandito che serve.
Ad esempio per l'alfabeto riportato in Figura 1.16, possiamo definire 8 stringhe, 4 per le lettere maiuscole e 4 per le minuscole, nelle quali ricercare, usando il codice ASCII di ogni lettera come indice, per trovare le 4 sottostringhe che lo descrivono. Segue come esempio il programma ES50.

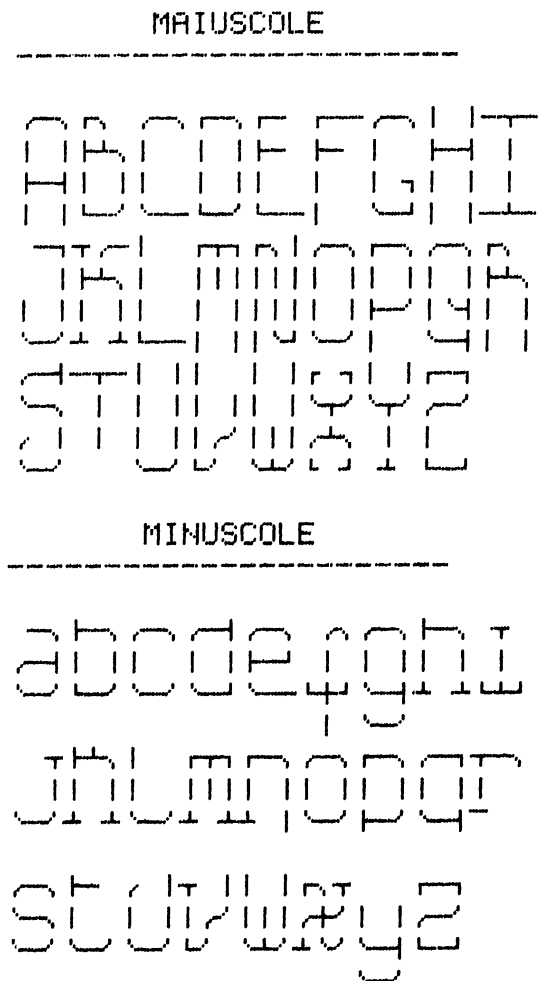



Figura 1.16 Alfabeto

```
1 REM ES50
5 REM ALFABETO
10 DIMA$(3),B$(3),C$(3)
11 Q$="■■■▧":W$="▶▶▛TTTT]"
13 FORJ=1TO3:FORK=0TO3:READD$:A$(K)=A$(K)+D$
15 NEXT:NEXT
18 FORJ=1TO3:FORK=0TO3:READD$:B$(K)=B$(K)+D$
20 NEXT:NEXT
28 PRINTCHR$(14);
30 INPUT"▧STRINGA:";S$:S$=LEFT$(S$,13)
35 PRINT"▧▧▧"CHR$(142);
40 FORI=1TOLEN(S$):CH=ASC(MID$(S$,I,1))
45 GOSUB300
50 FORJ=0TO3:PRINTC$(J)Q$;:NEXT:PRINTW$;:NEXTI
55 GETA$:IFA$<>"■"THEN55
60 GOTO28
300 REM SCRIVE CARATTERE
320 IFCH<65THEN380
325 IFCH>90THEN340
330 CH=CH-65
335 FORK=0TO3:C$(K)=MID$(B$(K),CH*3+1,3):NEXT
338 RETURN
340 IFCH<193THEN380
345 IFCH>218THEN380
350 CH=CH-193
355 FORK=0TO3:C$(K)=MID$(A$(K),CH*3+1,3):NEXT
358 RETURN
380 FORK=0TO3:C$(K)="   ":NEXT:RETURN
1000 REM MAIUSCOLE
1001 DATA"[graphics characters]"
1002 DATA"[graphics characters]"
1003 DATA"[graphics characters]"
1004 DATA"[graphics characters]"
1005 DATA"[graphics characters]"
1006 DATA"[graphics characters]"
1007 DATA"[graphics characters]"
1008 DATA"[graphics characters]"
1009 DATA"[graphics characters]"
1010 DATA"[graphics characters]"
1011 DATA"[graphics characters]"
1012 DATA"[graphics characters]"
```

```
1020 REM MINUSCOLE
1021 DATA"                                    "
1022 DATA"                                    "
1023 DATA"                                    "
1024 DATA"                                    "
1025 DATA"                                    "
1026 DATA"                                    "
1027 DATA"                                    "
1028 DATA"                                    "
1029 DATA"                                "
1030 DATA"                                "
1031 DATA"                                "
1032 DATA"                                "
```

Per poter stampare anche i numeri, bisogna prima preparare la descrizione riportata in Figura 1.17, quindi modificare il programma, come indicato in ES51, che non viene listato, ma è registrato sulla cassetta.

Figura 1.17 Cifre numeriche

Prova a modificare il programma per ottenere anche il carattere "@", oppure per ottenere altri caratteri speciali.

MESSAGGI SUL VIDEO: vogliamo realizzare scritte colorate che scorrono sul video. Ti proponiamo il programma ES52; in esso scriviamo i primi 40 caratteri della stringa e poi spostiamo il primo carattere in fondo alla stringa, metodo che risulta abbastanza veloce.

```
1 REM ES52
3 REM PAROLE CHE SCORRONO
5 S$="":FORK=1TO40:S$=S$+" ":NEXTK
10 INPUT"STRINGA1";A$:A$=S$+A$
20 INPUT"STRINGA2";B$:A$=A$+" "+B$
30 PRINT"[CLR][DOWN][DOWN]";
40 PRINTLEFT$(A$,40)"[UP]";:FORK=1TO100:NEXT
50 A$=MID$(A$,2)+LEFT$(A$,1):GOTO40
```

Puoi proporti di ottenere lo scorrimento in altre direzioni; buon lavoro!

OROLOGI: dato che il calcolatore misura il tempo trascorso dall'accensione usando 3 byte consecutivi in pagina 0, in 60-esimi di secondo, e che si può intervenire a modificare il valore iniziale dei byte, è possibile gestire un orologio. Per accedere da BASIC all'orologio interno del COMMODORE 64 sono disponibili le funzioni TI e TI$.
Segue il programma ES54 che visualizza l'orologio DIGITALE.

```
5 REM ES54
10 INPUT"[CLR]ORA (HHMMSS)";A$
15 IFLEN(A$)<>6THEN10
17 FORK=1TO6
19 IFMID$(A$,K,1)<"0"ORMID$(A$,K,1)>"9"THEN10
21 X1$=MID$(A$,1,2)
23 X2$=MID$(A$,3,2)
25 X3$=MID$(A$,5,2)
27 IFX1$>"23"ORX2$>"59"ORX3$>"59"THEN10
28 TI$=A$:PRINT"[CLR]";
30 FORT=0TO1:GOSUB100:T=0:NEXT
100 REM ORA CON [PRINT]
105 A$=TI$
110 PRINT"[HOME]"LEFT$(A$,2)":";
113 PRINTMID$(A$,3,2)":"RIGHT$(A$,2);
115 RETURN
```

1.7 COLLOQUIO SULLO SCHERMO

La parte fondamentale di un programma che non sia A SE STANTE è la interazione con l'utente. In alcuni programmi, specialmente nei videogiochi, questa interazione è resa il più semplice possibile, si preme un tasto, si sposta una levetta o una manopola e subito si può vedere sullo schermo l'effetto della propria azione. Questa è anche l'attuale tendenza negli ultimi modelli di calcolatori e nei più recenti programmi prodotti: l'utente deve ricordare il meno possibile; deve essere il programma a fornire una guida continua in modo che le operazioni siano semplici e prevedibili. Questo scopo viene raggiunto usando sofisticate tecniche di programmazione e spesso ricorrendo alla grafica raffinata. Qui ci limitiamo a vedere qualche esempio di interazione calcolatore-programma-utente ottenuta in MODO CARATTERI.

1.7.1 Il linguaggio naturale - i sottolinguaggi

Molto spesso, quando la velocità non è essenziale, si può ricorrere alla tastiera come principale dispositivo di ingresso dati (l'uscita è sempre sottinteso sia sul video) e i sistemi per garantire l'interattività sono di tipo conversazionale o guidati da menù. Nel primo caso si possono fare vari esempi. Lo stesso BASIC è di tipo conversazionale, anche se usa come possibili comandi solo un sottoinsieme dei verbi della lingua inglese.
Sarebbe sicuramente comodo poter impartire comandi come: CERCA IL VALORE MASSIMO DELLA FUNZIONE:..... TRA 0 e 10, oppure: CANCELLA LA REGISTRAZIONE DEL SIGNOR BIANCHI BRUNO. O anche, per risparmiare un pò sulla interpretazione, usare un linguaggio più telegrafico. Un esempio ben riuscito di questo tipo di interazione sono i cosiddetti ADVENTURE-GAME (giochi di avventura), nei quali il programma descrive una situazione in cui il giocatore si trova ed egli può dire al programma come decide di comportarsi, utilizzando un vocabolario alquanto ricco (purtroppo spesso solo in inglese, lingua che risulta molto adatta, dato che si basa su regole sintattiche abbastanza semplici per essere programmate).
Nella maggior parte degli altri casi il vocabolario viene ridotto ai soli verbi, seguiti da parametri numerici o alfabetici separati da barre, virgole o spazi, o altri caratteri separatori. Abbiamo già citato il BASIC, ma anche altri programmi di utilità generale usano queste tecniche.
Il più SCOMODO per l'utente, ma più semplice da programmare e più veloce da usare, è il sistema del riconoscimento della lettera INIZIALE, che fa riconoscere i comandi grazie al fatto che per distinguerli bastano la prima o le prime due lettere del verbo che li descrive (CA per CANCELLA e CE per CERCA). Anche se all'inizio può creare qualche perplessità, si impara rapidamente a usare questo metodo per fornire ordini ai programmi.

Inoltre, come ulteriore semplificazione, può essere preparata una specie di maschera da sovrapporre alla tastiera, che rechi un memo dei comandi. Alcuni programmi mostrano tale memo perennemente in una sezione del video.
Un sistema analogo è quello DOMANDA/RISPOSTA, utilizzabile quando il programma abbia un utilizzo specifico (operazioni sulle periferiche, procedure di calcolo,...); in questo caso il programma prende l'iniziativa cominciando a porre domande, a cui l'utente deve rispondere spesso solo con una parola o un numero in base al quale il programma continua a chiedere i successivi elementi di cui ha bisogno per poter compiere una determinata operazione.

1.7.2 I menù

E' un altro sistema diffusissimo per proporre delle scelte; esso consiste in un elenco delle operazioni possibili.

Un modo di presentare un menù è l'elencazione delle operazioni possibili una sotto l'altra, con a fianco il tasto da premere per la selezione. Basta premere il tasto giusto e il programma procede. Ovviamente non viene accettato un tasto non congruente.
Segue il programma esempio ES55.

```
1 REM ES55
5 REM SCELTA PER ELENCAZIONE
6 PRINT"▒▮▮"TAB(16)"M E N U´":PRINT
10 FORK=1TO5:READA$:PRINTTAB(10)A$:NEXTK
15 PRINT:PRINTTAB(10)"SCELTA >";
20 POKE198,0
30 GETA$:IFA$<"1"ORA$>"5"THEN30
40 PRINTA$;
50 POKE198,0
60 GETR$:IFR$=""THEN60
70 IFR$<>CHR$(13)THENPRINT"▮▮":GOTO30
75 PRINT
80 ONVAL(A$)GOTO100,200,300,400,500
100 PRINT"SCELTA N.1 ":STOP
200 PRINT"SCELTA N.2 ":STOP
300 PRINT"SCELTA N.3 ":STOP
400 PRINT"SCELTA N.4 ":STOP
500 PRINT"SCELTA N.5 ":STOP
1000 DATA"-OPERAZIONE 1 - 1 -"
```

```
1001 DATA"-OPERAZIONE 2 - 2 -"
1002 DATA"-OPERAZIONE 3 - 3 -"
1003 DATA"-OPERAZIONE 4 - 4 -"
1004 DATA"-OPERAZIONE 5 - 5 -"
```

Un altro modo è la presentazione di un elenco, con la possibilità di muoversi con un CURSORE, rappresentato da un particolare carattere, per puntare all'operazione desiderata. Trattiamo questo nel programma ES56.

```
 0 REM ES56
 3 S$="                                                  "
 5 PRINT"▣■"SPC(136)"M E N U´"SPC(32)"========"
 10 DIMA$(5):FORK=1TO5:READA$(K):NEXT
 15 PRINT:POKE53281,1
 20 FORK=1TO5:PRINTTAB(14)A$(K):PRINT:NEXT
 30 GOSUB100
 40 ONSGOTO50,60,70,80,90
 45 STOP
 50 PRINT"SCELTA N. 1":STOP
 60 PRINT"SCELTA N. 2":STOP
 70 PRINT"SCELTA N. 3":STOP
 80 PRINT"SCELTA N. 4":STOP
 90 PRINT"SCELTA N. 5":STOP
 95 DATA" 1 PRIMO   "," 2 SECONDO "
 96 DATA" 3 TERZO   "
 97 DATA" 4 QUARTO  "," 5 QUINTO  "
 100 REM SCEGLIE
 105 PRINT"▣"SPC(11)"[F1] SPOSTA IN GIU´";
 110 PRINTSPC(21)"[RETURN] CONFERMA"
 115 S=1
 120 PRINT"▣"SPC(120):FORK=1TOS
 123 PRINTSPC(80);:NEXT
 125 PRINTSPC(13)" ▬▬▬▬▬▬▬▬▬▬▬▬▬ "SPC(27);
 130 PRINT"▣■ "A$(S)"■ "SPC(27);
 135 PRINT" ▣▬▬▬▬▬▬▬▬▬▬▬▬■ "SPC(27)"┐┬┐";
 140 GETQ$:IFQ$<>CHR$(13)ANDQ$<>"■"THEN140
 145 PRINTS$" "A$(S)S$" ";
 150 IFQ$="■"THENS=S+1+5*(S=5):GOTO120
 155 PRINT"▣"S$S$;
 160 PRINT"▣"SPC(255)SPC(255)SPC(130);:RETURN
```

In certi casi tale CURSORE può essere rappresentato anche da un disegno abbastanza complesso che occupi più posizioni carattere. Molto bello è il modo di selezione presentato dal programma MAGIC DESK, nel quale si sposta una manina con l'indice puntato per mezzo di un Joystick.
Quello che è essenziale è che la selezione sia sicura e non possa dar luogo ad ambiguità di interpretazione.

Alcuni programmi abbastanza sofisticati consentono l'uso di un particolare tasto di HELP per richiamare un menù sul video in caso di bisogno, senza ovviamente perdere il precedente contenuto, che viene ripristinato quando non serve più il menù.

1.7.3 Come evidenziare una scelta

Una volta che l'utente ha operato una scelta, è bene che il programma lo informi che ha ricevuto, e, nei casi più delicati, chieda conferma. Il programma può evidenziare la scelta attivando il campo inverso, realizzando una sottolineatura, modificando il colore di una zona o altro.
L'importante è che l'utente si possa rendere conto di avere o non avere dato al programma l'ordine desiderato. Segue come esempio il programma ES57.

```
0 REM ES57
3 S$="                                        "
5 PRINT"♥■"SPC(96)"M E N U´"SPC(32)"========"
10 DIMA$(5):FORK=1TO5:READA$(K):NEXT
15 PRINT:POKE53281,1
20 FORK=1TO5:PRINTTAB(14)A$(K):PRINT:NEXT
30 GOSUB100
40 ONSGOTO50,60,70,80,90
45 STOP
50 PRINT"SCELTA N. 1":STOP
60 PRINT"SCELTA N. 2":STOP
70 PRINT"SCELTA N. 3":STOP
80 PRINT"SCELTA N. 4":STOP
90 PRINT"SCELTA N. 5":STOP
95 DATA" 1 PRIMO    "," 2 SECONDO  "
96 DATA" 3 TERZO    "
97 DATA" 4 QUARTO   "," 5 QUINTO   "
100 REM SCEGLIE
```

```
105 PRINT"▒"SPC(255)SPC(255)SPC(50)"SCELTA >";
110 GETQ$:IFQ$<"1"ORQ$>"5"THEN110
115 PRINTQ$"▒";:FORK=0TOVAL(Q$):PRINT"▒▒";:NEXT
120 PRINTSPC(54)"▒"A$(VAL(Q$))"▒";
125 PRINT"▒"SPC(10)"▒▒[ RETURN --> CONFERMA]";
130 PRINTSPC(18)"▒[ SPAZIO --> ANNULLA ]▒▒"
135 POKE198,0:S=VAL(Q$)
140 GETQ$:IFQ$<>" "ANDQ$<>CHR$(13)THEN140
145 PRINT"▒"S$S$SPC(255)SPC(225);
150 IFQ$=CHR$(13)THENPRINTS$:RETURN
155 PRINTSPC(8)" ";
160 PRINT"▒";:FORK=0TOS:PRINT"▒▒";:NEXT
165 PRINTSPC(54)A$(S);:GOTO105
```

CAPITOLO 2

LA GRAFICA

2.1 CAPACITA' GRAFICHE DEL COMMODORE 64

La grafica è una tra le più affascinanti applicazioni nel mondo dei calcolatori. Il tuo COMMODORE 64 possiede delle capacità grafiche talmente potenti da eguagliare quelle di calcolatori molto più costosi. Hai a disposizione:

- un video composto da ben 64000 puntini (pixel) disposti su 200 righe di 320 punti;
- la possibilità di sovrapporre all'immagine del video delle figure mobili, comodissime nelle animazioni grafiche;
- 16 bellissimi colori che ti aiutano a creare grafici di sicuro effetto.

In questo capitolo vedremo come definire caratteri diversi da quelli disponibili all'accensione e come gestire i 64000 punti che abbiamo a disposizione.
Nel Capitolo 3 vedremo come si possono usare le figure mobili (SPRITE).

2.2 LE OPERAZIONI LOGICHE AND E OR

Per poter parlare di grafica con il calcolatore è assolutamente necessario conoscere queste due operazioni logiche che sono essenziali anche in altre applicazioni del calcolatore.
Cominciamo col definire le operazioni AND e OR tra due bit:

0 AND 0 = 0   0 OR 0 = 0
0 AND 1 = 0   0 OR 1 = 1
1 AND 0 = 0   1 OR 0 = 1
1 AND 1 = 1   1 OR 1 = 1

Il risultato della AND tra due bit è quindi uguale a 1 solo se il primo E il secondo bit sono a 1; il risultato della OR tra due bit è uguale a 1 se il primo O il secondo bit (o entrambi) sono uguali a 1.
Ogni bit del risultato di una AND (o di una OR) tra bytes è dato dalla AND (o la OR) tra i bit corrispondenti degli operandi, ad esempio:

3 AND 2 = 2

infatti 3 in binario si esprime come 11 mentre 2 come 10 quindi, partendo dal bit meno significativo:

1 AND 0 = 0 e 1 AND 1 = 1

risultato 10, cioè 2 in decimale.

Ma come mai sono così importanti queste strane operazioni nella grafica col calcolatore? Perchè esse permettono di MASCHERARE alcuni bit nello scrivere o nel leggere un byte di memoria. Ad esempio, vogliamo sapere il contenuto del bit di posizione 4 (cioè il quinto da destra verso sinistra) di un certo byte? Bene, non dobbiamo far altro che fare l'operazione AND del byte con 16 (2 elevato a 4); se il risultato è uguale a 0 allora il bit è a zero, se è 16 (cioè 2 elevato a 4) allora il bit è a 1; infatti:

```
7 6 5 4 3 2 1 0 posizioni bit nel byte

? ? ? 0 ? ? ? ? (byte che vogliamo mascherare)
0 0 0 1 0 0 0 0 (16 in binario)
0 0 0 0 0 0 0 0 (0, risultato della AND)

? ? ? 1 ? ? ? ? (byte che vogliamo mascherare)
0 0 0 1 0 0 0 0 (16 in binario)
0 0 0 1 0 0 0 0 (16, risultato della AND)
```

Altro esempio: vogliamo porre a 1 il bit di posizione 3 di un byte, lasciando inalterati gli altri bit? Questa volta facciamo la OR tra il byte e 8 (2 elevato a 3); in questo modo il bit di posizione 3 del risultato diventa 1 (sia 0 che 1 OR 1 danno 1) e gli altri rimangono inalterati (1 OR 0 dà 1, 0 OR 0 dà 0).
Perdonaci ma, data l'importanza dell'argomento, vogliamo fare un ultimo esempio: dobbiamo porre a 0 il bit di posizione 2 di un certo byte. Come facciamo? Dobbiamo fare la AND tra questo byte e il numero 11111011 (255-4=251), infatti, in questo modo, il bit di posizione 2 verrà posto a 0 (sia 0 che 1 AND 0 danno 0) e gli altri bit rimarranno inalterati (0 AND 1 dà 0 ,1 AND 1 dà 1). Pensiamo ora un pò al nostro COMMODORE 64 e scriviamo il programma GRAF1.

```
1 REM GRAF1
10 N=4:M$="ESATTO !!!"
20 A=RND(0)
30 A=INT(RND(1)*2↑N)
40 B=INT(RND(1)*2↑N)
50 C=INT(RND(0)*2)
60 PRINT"QUANTO FA'"A;
70 D=A:GOSUB1000
80 PRINTTAB(20)B$
90 IFC=0THENPRINT"AND";:GOTO110
100 PRINT"OR";
110 PRINTB;
120 D=B:GOSUB1000
130 PRINTTAB(20)B$" ";
140 INPUTR
150 IFR=(AANDB)ANDC=0THENPRINTM$:PRINT:GOTO30
160 IFR=(AORB)ANDC=1THENPRINTM$:PRINT:GOTO30
170 PRINT"NO, IL RISULTATO E' ";
180 IFC=0THENR=AANDB:GOTO200
190 R=AORB
200 D=R:GOSUB1000
210 PRINTB$" ="R:PRINT:GOTO30
1000 B$=""
1010 FORI=N-1TO0STEP-1
1020 IF(DAND2↑I)=0THENB$=B$+"0":GOTO1040
1030 B$=B$+"1"
1040 NEXT
1050 RETURN
```

A cosa serve lo avrai già immaginato: ad allenarti con le operazioni AND e OR! Intanto vediamo come funziona, poi dai RUN al tuo calcolatore e buon divertimento!

COMMENTO A GRAF1.

Linea 10: pone uguale a 4 il numero di bit degli operandi: ti consigliamo di cominciare ad allenarti con 4 bit per poi passare ad operazioni a 8 bit (di più non servono per i nostri scopi); crea la costante M$.

Linea 20: sorteggia la base per i numeri random.

Linea 30: sceglie un numero intero tra 0 e 2 elevato a (N-1) come primo operando.

Linea 40: sceglie un numero intero tra 0 e 2 elevato a (N-1) come secondo operando.
Linea 50: sceglie fra 0 e 1; se 0 l'operazione proposta è AND, altrimenti l'operazione proposta è OR.
Linee 60-130: formula la domanda scrivendo gli operandi in decimale e in binario, usando la routine in 1000 che converte un numero decimale posto nella variabile D, nel corrispondente numero binario e lo pone nella string B$.
Linea 140: accetta la risposta.
Linee 150-160: controlla la risposta; se esatta lo segnala e riparte.
Linee 170-210: calcola e scrive la risposta esatta in decimale e binario e riparte.
Linea 1000: annulla la stringa che conterrà il numero binario.
Linee 1010-1040: calcola, partendo dal bit più significativo, il valore di ciascun bit facendo la AND tra il numero decimale e una maschera composta da tutti zeri tranne il bit che interessa, il risultato viene posto in B$.
Linea 1050: torna al programma principale.

2.3 LOCAZIONI GRAFICHE

Una delle caratteristiche del COMMODORE 64 che lo rende molto versatile nella programmazione grafica e ne permette una buona gestione della memoria, è il fatto che le locazioni grafiche (mappa video, mappa dei caratteri ecc.) non sono vincolate sempre alla stessa zona di memoria; infatti è possibile cambiarle scrivendo l'indirizzo che interessa negli appositi registri del processore video 6566/6567 (VIC II). Nei paragrafi seguenti abbiamo raccolto le spiegazioni delle operazioni necessarie a cambiare le locazioni grafiche. Nel seguito verranno richiamati, ogni volta che servono, i paragrafi che interessano. Se non hai già una certa conoscenza dei modi grafici del COMMODORE 64, ti consigliamo di continuare la lettura dal Paragrafo 2.4, e tornare a questi paragrafi quando trovi un richiamo.

2.3.1 Selezione del banco

Il bus indirizzi del VIC II è di soli 14 bit (può contenere al massimo il numero 16383), questo vuol dire che il VIC II può indirizzare solo 16 Kbyte di memoria alla volta. E' possibile selezionare quale dei 4 banchi di 16 Kbyte, presenti nel COMMODORE 64, debba essere VISTO dal VIC II. Il numero di banco è scritto, negato, nei due bit meno significativi del registro 0 (indirizzo 56576 (DD00H)) del secondo integrato di I/O, 6526 (CIA). Supponiamo, per esempio, che in quei due bit sia scritto il numero 01 (binario): negandolo otteniamo il numero 10, in questo caso quindi, il banco selezionato è il terzo; il numero B che distingue i banchi varia da 0 a 3. L'istruzione che serve per cambiare il banco è questa:

POKE 56576,(PEEK(56576)AND252)+N

Per trovare il valore desiderato di N consulta la Tabella 2.1.

TABELLA 2.1

BANCO	INDIRIZZI decimali	INDIRIZZI esadecimali	N
0	0-16383	0000H-3FFFH	3
1	16384-32767	4000H-7FFFH	2
2	32768-49151	8000H-BFFFH	1
3	49152-65535	C000H-FFFFH	0

Tabella 2.1 Valore di N per la selezione del banco

2.3.2 Memoria del video

L'indirizzo del primo byte della memoria video è dato da:

B*16384+V*1024

dove B è il banco selezionato (vedi Paragrafo 2.3.1) e V è il contenuto dei 4 bit più significativi del registro 24 del VIC II (indirizzo 53272 (D018H)). Il comando che serve per cambiare la locazione della memoria video è questo:

POKE53272,PEEK((53272)AND15)+16*N

Per trovare il valore desiderato di N consulta la Tabella 2.2.

TABELLA 2.2

LOCAZIONI DEL VIDEO decimali	LOCAZIONI DEL VIDEO esadecimali	N
0- 1023	0000H-03FFH	0
1024- 2047	0400H-07FFH	1
2048- 3071	0800H-0BFFH	2

3072- 4095	0C00H-0FFFH	3
4096- 5119	1000H-13FFH	4
5120- 6143	1400H-17FFH	5
6144- 7167	1800H-1BFFH	6
7168- 8191	1C00H-1FFFH	7
8192- 9215	2000H-23FFH	8
9216-10239	2400H-27FFH	9
10240-11263	2800H-2BFFH	10
11264-12287	2C00H-2FFFH	11
12288-13311	3000H-33FFH	12
13312-14335	3400H-37FFH	13
14336-15359	3800H-3BFFH	14
15360-16383	3C00H-3FFFH	15

Tabella 2.2 Valore di N per il posizionamento della memoria video

Ricorda che agli indirizzi scritti in tabella devi aggiungere il numero di banco moltiplicato per 16384 (4000H) (Paragrafo 2.3.1).
Nota che se cambi la locazione della mamoria video devi avvisare il SISTEMA OPERATIVO perchè sappia dove scrivere. Per far questo devi memorizzare V/256, nel byte 648 (288H), dove V è l'indirizzo del primo byte della memoria video.

2.3.3 Memoria del colore

La memoria del colore è formata da 1000 byte di memoria con solo 4 bit attivi (bastano infatti 4 bit per scegliere un colore tra 16 possibili); essa si trova dall'indirizzo 55296 (D800H) all'indirizzo 56295 (DBE7H) e non può essere spostata. E' da notare che questa zona di memoria fa parte dei 20 Kbyte DOPPI nel COMMODORE 64 (Paragrafo 8.5 volume sul BASIC). Per accedere alla RAM che risponde agli indirizzi della memoria del colore bisogna azzerare i 2 bit meno significativi del registro di I/O della CPU (indirizzo 1) disabilitando contemporaneamente le ROM del BASIC e del SISTEMA OPERATIVO. Questa operazione, che può esser fatta, ovviamente, solo da un programma in linguaggio macchina, esclude anche l'I/O, rendendo accessibile la RAM da 53248 a 57343 (D000H - DFFFH).

2.3.4 Memoria dei caratteri

L'indirizzo del primo byte della memoria dei caratteri è dato da:
B*16384+C*2048
dove B è il banco selezionato (Paragrafo 2.3.1) e C è il contenuto dei bit 3, 2 e 1 del registro 24 del VIC II (indirizzo 53272 (D018H)). Questa regola non è più valida in questi quattro casi:

1) B=0 e V=2
2) B=0 e V=3
3) B=2 e V=2
4) B=2 e V=3

In questi casi, infatti la memoria dei caratteri è la ROM generatrice. Poichè il bit di posizione 0 del registro 24 del VIC II non ha alcuna funzione, non lo devi mascherare quando cambi la locazione della memoria dei caratteri. Il comando che devi dare al tuo COMMODORE 64 sarà quindi:

POKE53272,PEEK((53272)AND240)+N

Per trovare il valore desiderato di N consulta la Tabella 2.3, nella quale, per maggior chiarezza, abbiamo scritto anche il numero B del banco.

TABELLA 2.3

LOCAZIONI DEI CARATTERI decimali	esadecimali	N	BANCO
0- 2047	0000H-07FFH	0	0
2048- 4095	0800H-0FFFH	2	0
ROM GENERATRICE		4	0
ROM GENERATRICE		6	0
8192-10239	2000H-27FFH	8	0
10240-12287	2800H-2FFFH	10	0
12288-14335	3000H-37FFH	12	0
14336-16383	3800H-3FFFH	14	0
16384-18431	4000H-47FFH	0	1
18432-20479	4800H-4FFFH	2	1
20480-22527	5000H-57FFH	4	1
22528-24575	5800H-5FFFH	6	1

24576-26623	6000H-67FFH	8	1
26624-28671	6800H-6FFFH	10	1
28672-30719	7000H-77FFH	12	1
30720-32767	7800H-7FFFH	14	1
32768-34815	8000H-87FFH	0	2
34816-36863	8800H-8FFFH	2	2
36864-38911	9000H-97FFH	4	2
ROM GENERATRICE		6	2
ROM GENERATRICE		8	2
43008-45055	A800H-AFFFH	10	2
45056-47103	B000H-B7FFH	12	2
47104-49151	B800H-BFFFH	14	2
49152-51199	C000H-C7FFH	0	3
51200-53247	C800H-CFFFH	2	3
53248-55295	D000H-D7FFH	4	3
55296-57343	D800H-DFFFH	6	3
57344-59391	E000H-E7FFH	8	3
59392-61439	E800H-EFFFH	10	3
61440-63487	F000H-F7FFH	12	3
63488-65535	F800H-FFFFH	14	3

Tabella 2.3 Valore di N per il posizionamento della memoria dei caratteri

Nota che il valore del bit di posizione 1 del registro 24 del VIC II viene cambiato ogni volta che premi i tasti COMMODORE e SHIFT insieme. Prova a eseguire:

PRINT PEEK(53272)

con il set di caratteri maiuscoli e poi:

print peek(53272)

con il set di caratteri minuscoli.

Vedrai che la risposta nel secondo caso è maggiore di due. Se questo fatto potesse provocarti dei problemi, nei tuoi programmi puoi inserire l'istruzione:

PRINT CHR$(8)

che non permetterà più di cambiare il set di caratteri da tastiera; per riabilitare SHIFT + CBM dai il comando PRINT CHR$ (9).

2.3.5 Memoria della pagina grafica

La memoria della pagina grafica è formata da 8000 byte di memoria. L'indirizzo della prima di queste locazioni è dato da B*16384+P*8192, dove B è il banco selezionato (vedi paragrafo 2.3.1) e P è il contenuto del bit 3 del registro 24 del VIC

II (indirizzo 53272 (D018H)). La pagina grafica, essendo così estesa, può essere messa solo in due posizioni diverse per ogni banco: i primi 8000 byte o i primi 8000 byte della seconda metà.
Ovviamente, nel banco 0 non è possibile porre la pagina grafica nella prima posizione, perchè verrebbe a sovrapporsi alla pagina 0; inoltre nella prima posizione del banco 2 e del banco 0 il VIC VEDE tra gli indirizzi 38912 e 43007 (9800H-A7FFH) la ROM generatrice dei caratteri (vedi paragrafo precedente). Per cambiare indirizzo della pagina grafica le istruzioni sono:

POKE 53272,PEEK(53272)AND247 per la prima posizione nel banco,

POKE 53272,PEEK(53272)OR8 per la seconda posizione nel banco.

Nota che per cambiare posizione della pagina grafica si usa un bit che interessa anche la posizione della memoria dei caratteri: quando torni al modo caratteri devi quindi ricordarti di porre in quel bit il valore corretto.

2.3.6 Sovrapposizione di un area di memoria grafica alla ROM

Come avrai notato il COMMODORE 64 ti dà la possibilità di porre la memoria video, la memoria dei caratteri e la memoria della pagina grafica in locazioni in cui rispondono le ROM del BASIC e del SISTEMA OPERATIVO. Cosa succede in tal caso? Il VIC II, che può solamente leggere la memoria, legge la RAM che risponde a quegli indirizzi e non sicuramente la ROM (non può interessare al VIC II il contenuto della ROM del BASIC o del SISTEMA OPERATIVO); alla CPU, invece, interessano i contenuti di queste ROM, per cui, quando accede a quegli indirizzi per leggere, legge la ROM, ma quando scrive sa di non riuscire a scrivere su una ROM, quindi scrive sulla RAM. Come vedi, anche se allochi un'area di memoria grafica in indirizzi dedicati alle ROM, tutto va bene perchè la CPU riesce a scrivere e il VIC II riesce a leggere. Usare un'area di questo tipo comporta il vantaggio che non si RUBA memoria al programma e alle variabili BASIC, ma non si riesce a sapere il contenuto dei byte di RAM poichè la CPU non riesce a leggerli. Si può risolvere questo piccolo problema con una routine in linguaggio macchina che disabiliti la ROM interessata (deve anche trascurare gli interrupt) e permetta alla CPU di leggere la RAM.

2.4 CARATTERI DEFINIBILI

Tu sai già come il COMMODORE 64 possa visualizzare tutti i caratteri che puoi vedere sul video, e sai anche che le IMMAGINI di questi caratteri (una serie di 8 byte per ogni carattere) sono gelosamente custodite in ROM (memoria a sola lettura), perchè non vengano perse ogni volta che spegni il calcolatore. Può sembrare, quindi che ogni volta che verrà premuto il tasto "A" del COMMODORE 64 il video mostrerà una A. Invece il COMMODORE 64 ti dà anche la possibilità di creare uno per uno tutti i caratteri. Proviamo a definire un carattere: per esempio il simbolo CBM che appare sul tasto in basso a sinistra della tastiera del tuo calcolatore. Usando la stessa tecnica, con cui sono disegnati normalmente i caratteri, costruiamo una matrice 8 per 8 e riempiamola in modo da ottenere il simbolo voluto, usando 0 per i punti spenti e 1 per quelli accesi:

```
0 0 0 1 1 0 0 0
0 0 1 1 1 0 0 0
0 1 1 0 0 1 1 1
0 1 1 0 0 1 1 0
0 1 1 0 0 1 1 1
0 0 1 1 1 0 0 0
0 0 0 1 1 0 0 0
0 0 0 0 0 0 0 0
```

Ora ci tocca fare un pò di conti: la prima riga (cioè il primo byte del carattere) contiene il numero binario 00011000, cioè 24 decimale. Facendo lo stesso calcolo per le altre righe otteniamo una serie di 8 numeri cioè 24, 56, 103, 102, 103, 56, 24, 0..
Ci basta ora cambiare al VIC II l'indirizzo della mappa dei caratteri (Paragrafi 2.3.1 e 2.3.4) e memorizzare gli 8 numeri ottenuti nei primi 8 byte della nuova mappa dei caratteri, programmando il primo carattere, quello che corrisponde al D-CODE 0, e che all'accensione rappresenta la chiocciola (Paragrafo 7.4 del volume sul BASIC).
Nota che, creando il carattere, abbiamo cercato di non usare mai uno 0 o un 1 ISOLATO (cioè con un simbolo opposto sia a destra che a sinistra): infatti se usi un televisore o un monitor di qualità non ottima, è facile che, non facendo così, appaiano sullo schermo effetti non desiderati. Dopo aver capito bene il meccanismo di programmazione dei caratteri, prova a creare un carattere contenente in tutti i byte il numero 85 ,cioè 01010101 binario, e guarda se crea strani effetti sul tuo monitor.
Proviamo ora il programma GRAF2.

```
1 REM GRAF2
10 POKE53280,0:POKE53281,0
15 POKE53272,(PEEK(53272)AND240)+12
20 FORI=0TO7:READA:POKE12*1024+I,A:NEXT
30 PRINTCHR$(154)CHR$(142)CHR$(147)"@"
40 GETA$:IFA$=""THEN40
50 POKE53272,(PEEK(53272)AND240)+5
60 DATA24,56,103,102,103,56,24,0
```

Fai girare il programma: vedrai sullo schermo, in alto a sinistra il carattere che abbiamo appena programmato. Se premi un tasto tutto torna normale e vedrai che il carattere torna ad essere quello che è normalmente: una chiocciola. Se invece di premere un tasto qualunque premi il tasto di STOP e provi a scrivere qualche cosa, lo schermo si riempirà di caratteri incomprensibili (non ti preoccupare, premendo STOP-RESTORE tutto tornerà alla normalità); questi caratteri sono tutti i caratteri che non abbiamo programmato, infatti ci siamo preoccupati di riempire solo i primi 8 byte della nuova mappa.

COMMENTO A GRAF2.
Linea 10: pone schermo e bordo neri e scrive 12 nel nibble (4 bit) meno significativo del registro 24 del VIC II, senza cambiare il valore degli altri 4 bit (sposta la mappa dei caratteri a 12288 (3000H)).
Linea 20: riempe la matrice del primo carattere (la chiocciola) con i dati che abbiamo calcolato prima.
Linea 30: pulisce lo schermo e scrive in blù chiaro, nel set maiuscolo, una chiocciola. (CHR$(8) disabilita la funzione dei tasti COMMODORE-SHIFT).
Linea 40: attende che sia premuto un tasto.
Linea 50: scrive 5 nel nibble meno significativo del registro 24 del VIC II, senza cambiare il valore degli altri 4 bit (sposta nuovamente la mappa dei caratteri agli indirizzi della ROM).
Linea 60: contiene gli 8 dati che formano il carattere.

2.4.1 Copia dei caratteri dalla ROM

Non sempre, quando definisci un nuovo set di caratteri, vuoi ottenere un risultato completamente diverso dal contenuto della ROM: potrebbe capitarti di volere avere a disposizione tutte le lettere del set del COMMODORE 64 e cambiarne solo i simboli grafici, o tenere i caratteri POSITIVI (normali) e cambiare quelli in campo inverso. Cosa fare in questo caso? Ovviamente dovrai copiare su RAM la parte del

contenuto della ROM che ti interessa e, quindi, definire come abbiamo già visto gli altri caratteri. L'unico problema è dato dal fatto che la CPU non VEDE, normalmente, la ROM generatrice di caratteri. Per far si che la CPU possa leggere da questa ROM bisogna porre a zero il bit di posizione 2 del registro di I/O della CPU stessa (chè risponde all'indirizzo 1), eseguendo il comando:
POKE 1, PEEK(1) AND 251.
Purtroppo questa operazione disabilita l' I/O cioè i due CIA (gli integrati di I/O), il SID (l'integrato del suono) e il VIC II poichè la ROM generatrice risponde proprio agli indirizzi (53248-57343 (D000-DFFF)) cui rispondono i registri di questi integrati. Non è quindi conveniente tenere sempre inserita la ROM a scapito di tutti gli integrati di I/O. Poichè, inoltre, la routine di interrupt ha bisogno di accedere alle locazioni di I/O, per poter trasferire il contenuto della ROM dei caratteri in RAM, dovremo anche disabilitare l'interrupt eseguendo l'istruzione:
POKE 56334, PEEK (56334) AND 254
cioè ponendo a zero il bit di posizione 0 del registro 14 del CIA 1 che ferma il contatore che genera l'interrupt. Prova ora il programma GRAF3.

```
1 REM GRAF3
5 PRINTCHR$(142)
10 POKE56,48:POKE55,0:CLR
20 A=PEEK(56334):POKE56334,AAND254
30 POKE1,PEEK(1)AND251
40 FORI=0TO2047:POKEI+12288,PEEK(I+53248):NEXT
50 POKE1,PEEK(1)OR4
60 POKE56334,A
70 POKE53272,(PEEK(53272)AND240)+12
```

Il programma occuperà il tuo COMMODORE 64 per circa 35 secondi, e quando avrà finito dirà semplicemente "READY". Nulla sembrerà essere successo, ma in realtà ora i dati relativi ai caratteri non vengono più dalla ROM ma dalla RAM. Prova infatti a scrivere una chiocciola sullo schermo e poi esegui il comando:
FOR I=0 TO 7:POKE 12288+I,0:NEXT
appena premi RETURN la chiocciola sparirà dallo schermo e non sarà più possibile scriverne un'altra poichè hai appena cancellato i dati relativi alla chiocciola in RAM. Prova poi a selezionare il set delle minuscole e vedrai che quei caratteri non sono stati programmati.

COMMENTO A GRAF3.
Linea 5: seleziona il set delle maiuscole.
Linea 10: pone la fine della memoria riservata al BASIC all'indirizzo 12288 (3000H)

perchè il programma e le variabili non sporchino i caratteri programmati che partiranno, appunto, dall'indirizzo 12288.
Linea 20: pone il contenuto del registro 14 del CIA 1 nella variabile A e disabilita l'interrupt.
Linea 30: esclude l'I/O per abilitare la ROM dei caratteri.
Linea 40: trasferisce metà dei caratteri della ROM (il set delle maiuscole: dall'indirizzo 53248 all'indirizzo 55295) in RAM a partire dall'indirizzo 12288.
Linea 50: esclude la ROM riabilitando l'I/O.
Linea 60: pone nel registro 14 del CIA 1 il valore iniziale per riabilitare l'interrupt.
Linea 70: pone l'inizio della mappa dei caratteri in 12288.

Prova ora a scrivere il programma GRAF4 il cui compito è proprio quello di creare un nuovo set di caratteri che sia uguale a quello in ROM per i caratteri normali, e diverso per quelli in campo inverso.

```
1 REM GRAF4
5 PRINTCHR$(142)
10 POKE56,48:POKE55,0:CLR:S=111713:NC=128
17 M$=" ECCO I NUOVI CARATTERI DEL COMMODORE !"
20 A=PEEK(56334):POKE56334,AAND254
30 POKE1,PEEK(1)AND251
40 FORI=0TO1023:POKEI+12288,PEEK(I+53248):NEXT
50 POKE1,PEEK(1)OR4
60 POKE56334,A
70 FORI=1TONC*8
80 READA:B=B+A
90 POKEI+13311,A
100 NEXTI
105 IFB<>STHENPRINT"ERRORE NELLE LINEE DATA":STOP
110 PRINTCHR$(147)
120 POKE53272,(PEEK(53272)AND240)+12
130 PRINTCHR$(17)CHR$(18)M$
1000 DATA120,254,238,238,232,224,254,126
1010 DATA56,124,238,254,254,238,238,238
1020 DATA252,238,238,252,238,238,254,252
1030 DATA124,254,230,224,224,238,254,124
1040 DATA248,252,238,238,238,238,252,248
1050 DATA254,254,224,248,248,224,254,254
1060 DATA254,254,224,248,248,224,224,224
1070 DATA62,126,224,238,238,230,126,62
```

```
1080 DATA238,238,238,254,254,238,238,238
1090 DATA254,254,56,56,56,56,254,254
1100 DATA254,254,56,56,56,248,248,48
1110 DATA238,236,252,248,248,236,238,238
1120 DATA224,224,224,224,224,224,254,254
1130 DATA198,238,254,254,254,214,198,198
1140 DATA198,230,246,254,254,222,206,198
1150 DATA124,254,238,238,238,238,254,124
1160 DATA252,254,230,254,252,224,224,224
1170 DATA124,238,238,238,238,254,124,14
1180 DATA252,230,230,254,248,252,238,230
1190 DATA124,254,224,252,126,14,254,124
1200 DATA254,254,56,56,56,56,56,56
1210 DATA238,238,238,238,238,238,254,124
1220 DATA238,238,238,238,238,254,124,56
1230 DATA198,198,214,254,254,254,238,198
1240 DATA238,238,124,56,56,124,238,238
1250 DATA238,238,238,124,56,56,56,56
1260 DATA254,254,30,60,120,240,254,254
1270 DATA124,120,112,112,112,112,120,124
1280 DATA15,31,48,255,255,112,255,255
1290 DATA62,30,14,14,14,14,30,62
1300 DATA0,24,60,126,24,24,24,24
1310 DATA0,16,48,127,127,48,16,0
1320 DATA0,0,0,0,48,48,0,0
1330 DATA60,60,60,60,60,0,60,60
1340 DATA238,238,238,0,0,0,0,0
1350 DATA102,255,255,102,255,255,102,0
1360 DATA24,254,224,255,7,127,124,24
1370 DATA227,231,238,28,56,119,231,199
1380 DATA60,102,62,120,239,230,127,61
1390 DATA14,28,56,96,0,0,0,0
1400 DATA60,120,240,240,240,240,120,60
1410 DATA60,30,15,15,15,15,30,60
1420 DATA24,90,60,255,60,90,24,0
1430 DATA0,56,56,254,254,56,56,0
1440 DATA0,0,0,0,0,24,56,112
1450 DATA0,0,0,126,126,0,0,0
1460 DATA0,0,0,0,0,56,56,56
1470 DATA3,7,14,28,56,112,224,192
1480 DATA124,254,238,238,238,254,254,124
1490 DATA56,56,120,120,56,56,254,254
```

```
1500 DATA124,254,134,12,56,96,254,254
1510 DATA124,254,14,60,60,14,254,124
1520 DATA28,60,124,204,254,254,12,12
1530 DATA254,224,252,14,14,206,254,124
1540 DATA124,238,224,252,238,238,254,124
1550 DATA254,238,28,56,56,56,56,56
1560 DATA124,238,238,124,238,238,254,124
1570 DATA124,238,238,126,14,206,254,124
1580 DATA0,56,56,0,0,56,56,0
1590 DATA0,0,56,56,0,56,56,112
1600 DATA2,14,62,254,254,62,14,2
1610 DATA0,0,126,126,0,126,126,0
1620 DATA64,112,124,127,127,124,112,64
1630 DATA60,126,102,14,28,28,0,28
1640 DATA0,0,0,255,255,0,0,0
1650 DATA8,28,62,127,127,28,62,0
1660 DATA24,24,24,24,24,24,24,24
1670 DATA0,0,0,255,255,0,0,0
1680 DATA0,0,255,255,0,0,0,0
1690 DATA0,255,255,0,0,0,0,0
1700 DATA0,0,0,0,255,255,0,0
1710 DATA48,48,48,48,48,48,48,48
1720 DATA12,12,12,12,12,12,12,12
1730 DATA0,0,0,224,240,56,24,24
1740 DATA24,24,28,15,7,0,0,0
1750 DATA24,24,56,240,224,0,0,0
1760 DATA192,192,192,192,192,192,255,255
1770 DATA192,224,112,56,28,14,7,3
1780 DATA3,7,14,28,56,112,224,192
1790 DATA255,255,192,192,192,192,192,192
1800 DATA255,255,3,3,3,3,3,3
1810 DATA0,60,126,126,126,126,60,0
1820 DATA0,0,0,0,0,255,255,0
1830 DATA54,127,127,127,62,28,8,0
1840 DATA96,96,96,96,96,96,96,96
1850 DATA0,0,0,7,15,28,24,24
1860 DATA195,231,126,60,60,126,231,195
1870 DATA0,60,126,102,102,126,60,0
1880 DATA24,24,102,102,24,24,60,0
1890 DATA6,6,6,6,6,6,6,6
1900 DATA8,28,62,127,62,28,8,0
1910 DATA24,24,24,255,255,24,24,24
```

```
1920 DATA192,192,48,48,192,192,48,48
1930 DATA24,24,24,24,24,24,24,24
1940 DATA0,0,3,62,118,54,54,0
1950 DATA255,127,63,31,15,7,3,1
1960 DATA0,0,0,0,0,0,0,0
1970 DATA240,240,240,240,240,240,240,240
1980 DATA0,0,0,0,255,255,255,255
1990 DATA255,255,0,0,0,0,0,0
2000 DATA0,0,0,0,0,0,0,255
2010 DATA192,192,192,192,192,192,192,192
2020 DATA204,204,51,51,204,204,51,51
2030 DATA3,3,3,3,3,3,3,3
2040 DATA0,0,0,0,204,204,51,51
2050 DATA255,254,252,248,240,224,192,128
2060 DATA3,3,3,3,3,3,3,3
2070 DATA24,24,24,31,31,24,24,24
2080 DATA0,0,0,0,15,15,15,15
2090 DATA24,24,24,31,31,0,0,0
2100 DATA0,0,0,248,248,24,24,24
2110 DATA0,0,0,0,0,0,255,255
2120 DATA0,0,0,31,31,24,24,24
2130 DATA24,24,24,255,255,0,0,0
2140 DATA0,0,0,255,255,24,24,24
2150 DATA24,24,24,248,248,24,24,24
2160 DATA192,192,192,192,192,192,192,192
2170 DATA224,224,224,224,224,224,224,224
2180 DATA7,7,7,7,7,7,7,7
2190 DATA255,255,0,0,0,0,0,0
2200 DATA255,255,255,0,0,0,0,0
2210 DATA0,0,0,0,0,255,255,255
2220 DATA3,3,3,3,3,3,255,255
2230 DATA0,0,0,0,240,240,240,240
2240 DATA15,15,15,15,0,0,0,0
2250 DATA24,24,24,248,248,0,0,0
2260 DATA240,240,240,240,0,0,0,0
2270 DATA0,0,255,255,0,0,0,8
```

COMMENTO A GRAF4.

Le linee dalla 5 alla 60 sono uguali a quelle del programma GRAF3 che abbiamo appena visto, tranne che nei seguenti particolari: nella linea 10 inizializziamo le variabili S e NC che servono per fare un controllo sulle linee DATA (S=somma di

tutti i valori contenuti nei data, NC=numero di caratteri che si vogliono programmare); nella linea 40 il loop finisce a 1023 invece che a 2047 perchè non copiamo i caratteri in campo inverso.
Linea 70: inizializza un loop di NC*8 cicli (di lunghezza diversa in dipendenza dal numero di caratteri che si vogliono programmare).
Linea 80: legge uno dopo l'altro i dati delle linee DATA e ne fa la somma ponendola nella variabile B.
Linea 90: mette i dati letti nei byte da 13312 in poi (parte che sarà dedicata ai caratteri in campo inverso).
Linea 100: chiude il loop.
Linea 105: controlla che la somma sia uguale alla variabile S.
Linea 110: pulisce lo schermo.
Linea 120: pone l'inizio della mappa dei caratteri in 12288.
Linea 130: scrive, una linea più in basso, in campo inverso (cioè con il nuovo set di caratteri).
Linee 1000-2270: 128 linee di 8 dati ciascuna: ogni linea contiene i dati relativi ad ognuno dei 128 caratteri in campo inverso.

Copiare questo programma, con 128 linee DATA, può essere piuttosto monotono: abbiamo quindi calcolato il valore che S deve avere se si programmano solo 32, 64 o 96 caratteri. Per NC=32, S vale 47284; per NC=64, S vale 72982 e per NC=96, S vale 90667. Puoi così programmare solo 32 caratteri e, se il risultato è di tuo gradimento, programmare gli altri in un secondo tempo, o a più riprese.

2.4.2 Programmi per la creazione dei caratteri

Programmare nuovi caratteri con il COMMODORE 64 è facile ma non proprio molto immediato: è difficile cioè VEDERE che 24, 56, 103, 102, 103, 56, 24, 0 rappresentano un simbolo grafico. Sarebbe molto più comodo se si potesse disegnare direttamente sullo schermo il carattere e farlo memorizzare senza dover fare tanti calcoli, PEEK e POKE. Il programma GRAF5 fà precisamente questo. E' commentato come al solito, e puoi quindi eliminare certe parti che non ti interessano e magari aggiungerne altre per renderlo più adatto ai tuoi scopi.

```
1 REM GRAF5
10 POKE13*4096+32,0:POKE13*4096+33,0
15 PRINTCHR$(154)CHR$(142)CHR$(8)
20 POKE56,48:POKE55,0:CLR
```

```
25 US$="              PREMI F1 PER USCIRE"
26 US$=CHR$(147)+US$+CHR$(31)
100 PRINTCHR$(147)
110 PRINT"OPZIONI  :":PRINT:PRINT
120 PRINT"1 CREA CARATTERE":PRINT
130 PRINT"2 MEMORIZZA CARATTERE":PRINT
140 PRINT"3 CORREGGE CARATTERE":PRINT
150 PRINT"4 MOSTRA CARATTERI":PRINT
160 PRINT"5 SALVA SET DI CARATTERI":PRINT
170 PRINT"6 CARICA SET DI CARATTERI":PRINT
180 PRINT"7 FINE"
190 GETA$:IFVAL(A$)=0THEN190
195 I=VAL(A$)
200 ONIGOSUB1000,2000,3000,4000,5000,6000,7000
210 GOTO100
1000 PRINTUS$
1010 FORI=1TO8:PRINTCHR$(17);:NEXTI
1020 FORI=1TO8
1030 FORJ=1TO16:PRINTCHR$(32);:NEXT
1040 FORJ=1TO8:PRINTCHR$(43);:NEXTJ
1050 PRINT:NEXTI
1060 PC=55672:XC=0:YC=0
1070 POKEPC,1
1080 GETA$:IFA$=""THEN1080
1090 A=ASC(A$):SP=0
1100 IFA=29ANDXC<7THENXC=XC+1:PC=PC+1:SP=1
1110 IFA=157ANDXC>0THENXC=XC-1:PC=PC-1:SP=-1
1120 IFA=17ANDYC<7THENYC=YC+1:PC=PC+40:SP=40
1130 IFA=145ANDYC>0THENYC=YC-1:PC=PC-40:SP=-40
1140 IFA=32THENPOKEPC-54272,160
1150 IFA=20THENPOKEPC-54272,43
1160 IFA=133THENPRINTCHR$(154):POKEPC-SP,6
1165 IFA=133THEN1180
1170 POKEPC-SP,6:GOTO1070
1180 FORI=0TO7
1190 BY(I)=0
1200 FORJ=0TO7
1210 BY(I)=BY(I)-2↑J*(PEEK(1400+7-J+40*I)=160)
1220 NEXTJ:NEXTI
1230 RETURN
2000 PRINTCHR$(147)
2010 PRINT"1 NORMALE":PRINT
```

```
2020 PRINT"2 REVERSATO":PRINT
2030 PRINT"3 SIMMETRIA VERTICALE":PRINT
2040 PRINT"4 SIMMETRIA ORIZZONTALE":PRINT
2050 PRINT"5 RUOTATO DI 90 GRADI ";
2055 PRINT"IN SENSO ORARIO":PRINT
2060 GETA$:IFA$=""THEN2060
2070 A=VAL(A$):IFA=0ORA>5THEN2060
2080 INPUT"CARATTERE NUMERO ";NC%
2090 IFNC%>511ORN%<0THEN2080
2100 ONAGOSUB2200,2300,2400,2500,2600
2110 FORI=0TO7:POKE12288+8*NC%+I,BY(I):NEXT
2120 RETURN
2200 RETURN
2300 FORI=0TO7:BY(I)=255-BY(I):NEXT
2310 RETURN
2400 FORI=0TO7
2410 FORJ=0TO3
2420 BD=(BY(I)AND2↑J)/2↑J:BY(I)=BY(I)-2↑J*BD
2430 BS=(BY(I)AND2↑(7-J))/2↑(7-J)
2435 BY(I)=BY(I)-2↑(7-J)*BS
2440 BY(I)=BY(I)+2↑J*BS:BY(I)=BY(I)+2↑(7-J)*BD
2450 NEXTJ,I
2460 RETURN
2500 FORI=0TO3
2510 T=BY(I)
2520 BY(I)=BY(7-I)
2530 BY(7-I)=T
2540 NEXTI
2550 RETURN
2600 FORI=0TO7
2610 B2(I)=BY(I):BY(I)=0
2620 NEXTI
2630 FORI=0TO7
2640 FORJ=0TO7
2645 EX=I*((B2(I)AND2↑(7-J))/2↑(7-J))
2650 BY(J)=BY(J)OR2↑EX
2660 NEXTJ,I
2670 RETURN
3000 PRINTCHR$(147)
3010 INPUT"CARATTERE DA CORREGGERE ";NC%
3020 IFNC%>511ORN%<0THEN3010
3030 PRINTUS$
```

```
3040 FORI=1TO8:PRINTCHR$(17);:NEXTI
3050 FORI=0TO7
3060 FORJ=1TO16:PRINTCHR$(32);:NEXT
3070 FORJ=0TO7
3080 IFPEEK(12288+8*NC%+I)AND2↑(7-J)THENSI=-1
3085 IFSITHENPRINTCHR$(18)CHR$(32)CHR$(146);
3086 IFSITHENSI=0:GOTO3100
3090 PRINTCHR$(43);
3100 NEXTJ:PRINT:NEXTI
3110 GOTO1060
4000 PRINTCHR$(147)"PREMI F1 PER TORNARE"
4005 PRINTCHR$(17)"PREMI I TASTI CORRISPONDENTI"
4006 PRINT"AI CARATTERI CHE VUOI VEDERE"
4010 FORI=1TO2000:NEXTI
4020 PRINTCHR$(147);
4030 POKE53272,(PEEK(53272)AND240)+12
4040 GETA$:IFA$=""THEN4040
4045 IFA$=CHR$(133)THENF1=-1
4050 IFF1THENPOKE53272,(PEEK(53272)AND240)+5
4055 IFF1THENF1=0:PRINTCHR$(154):RETURN
4060 PRINTA$;
4070 GOTO4040
5000 PRINTCHR$(147)
5010 PRINT"DISCO O NASTRO ?"
5015 GETDV$:IFDV$<>"N"ANDDV$<>"D"THEN5015
5020 PRINT
5030 INPUT"SET NUMERO ";NS%
5040 PRINT
5050 PRINT"FINO A CHE CARATTERE VUOI SALVARE ";
5055 INPUTNC%
5060 IFNC%>255THEN5040
5070 IFDV$="N"THEN5500
5080 OPEN1,8,2,"@:SET #"+STR$(NS%)+",S,W"
5082 GOSUB9000
5085 IFNNTHENFORI=1TO2000:NEXT:CLOSE1:CLOSE15
5087 IFNNRETURN
5090 PRINT#1,CHR$(NC%);:FORI=0TO(NC%+1)*8-1
5100 PRINT#1,CHR$(PEEK(12288+I));
5110 NEXTI
5120 CLOSE1:CLOSE15
5130 RETURN
```

```
5500 OPEN1,1,1,"SET #"+STR$(NS%)
5510 PRINT#1,CHR$(NC%);:FORI=0TO(NC%+1)*8-1
5520 PRINT#1,CHR$(PEEK(12288+I));
5530 NEXTI
5540 CLOSE1
5550 RETURN
6000 PRINTCHR$(147)
6010 PRINT"DISCO O NASTRO ?"
6015 GETDV$:IFDV$<>"N"ANDDV$<>"D"THEN6015
6020 PRINT
6030 INPUT"SET NUMERO ";NS%
6070 IFDV$="N"THEN6500
6080 OPEN1,8,2,"SET #"+STR$(NS%)+",S,R"
6082 GOSUB9000
6085 IFNNTHENFORI=1TO2000:NEXT:CLOSE1:CLOSE15
6087 IFNNTHENRETURN
6090 GET#1,A$:NC%=ASC(A$):FORI=0TO(NC%+1)*8-1
6100 GET#1,A$:POKE12288+I,ASC(A$+CHR$(0))
6110 NEXTI
6120 CLOSE1:CLOSE15
6130 RETURN
6500 OPEN1,1,0,"SET #"+STR$(NS%)
6510 FORI=0TO(NC+1)%*8-1
6520 GET#1,A$:POKE12288+I,ASC(A$+CHR$(0))
6530 NEXTI
6540 CLOSE1
6550 RETURN
7000 PRINTCHR$(17)"SICURO ?"
7010 GETA$:IFA$=""THEN7010
7020 IFA$="S"THENSYS58260
7030 RETURN
9000 PRINT:OPEN15,8,15:INPUT#15,NN,MM$,TT,SS
9005 PRINTNN,MM$,TT,SS
9010 RETURN
```

Il programma GRAF5 può essere diviso in 15 sezioni ben distinte:

- 1: le linee 10-26 inizializzano il calcolatore.
- 2: le linee dalla 100 alla 210 mostrano il menù e saltano alla funzione richiesta.
- 3: le linee dalla 1000 alla 1230 ti permettono di disegnare un nuovo carattere e riempiono un vettore con i dati relativi al carattere disegnato (i calcoli li fa il programma).

- 4: le linee dalla 2000 alla 2120 mostrano il menù di memorizzazione dei caratteri, saltano alla routine per il tipo di operazione richiesta sul vettore e memorizzano il carattere.
- 5: la linea 2200 lascia il vettore come è; è stata messa per rispondere alla chiamata della linea 2100.
- 6: le linee dalla 2300 alla 2310 pongono nel vettore il NEGATIVO del carattere già contenuto.
- 7: le linee dalla 2400 alla 2460 pongono nel vettore il carattere che gode di simmetria verticale rispetto a quello già contenuto.
- 8: le linee dalla 2500 alla 2550 pongono nel vettore il carattere che gode di simmetria orizzontale rispetto a quello già contenuto.
- 9: le linee dalla 2600 alla 2670 pongono nel vettore il carattere già contenuto, ruotato di 90 gradi in senso orario. Chiamando tre volte questa routine otterrai nel vettore il carattere ruotato di 90 gradi in senso antiorario.
- 10: le linee dalla 3000 alla 3110 riempono il vettore con i dati relativi al carattere che vuoi correggere e, saltando alla linea 1060, ti permettono di correggere questo carattere.
- 11: le linee dalla 4000 alla 4070 ti permettono di vedere i caratteri che hai creato.
- 12: le linee dalla 5000 alla 5550 salvano su disco o su nastro i caratteri che hai programmato, assegnando al file un nome di riconoscimento.
- 13: le linee dalla 6000 alla 6550 caricano da disco o da nastro un set di caratteri salvato precedentemente.
- 14: le linee dalla 7000 alla 7030 chiudono il programma.
- 15: le linee dalla 9000 alla 9010 scrivono lo stato del disco.

COMMENTO A GRAF5 SEZIONE PER SEZIONE.

SEZIONE 1.

Linee 10-15: pone schermo e sfondo neri, il colore del cursore blu chiaro, seleziona il set delle maiuscole e disabilita la funzione dei tasti CBM e SHIFT.
Linea 20: pone i puntatori di fine memoria a 12288 (3000H) per non sporcare con le variabili i caratteri che verranno creati.
Linee 25-26: inizializza la costante US$.

SEZIONE 2.

Linee 100-180: mostra le opzioni.
Linea 190: accetta un carattere numerico compreso tra 1 e 7.
Linea 195: pone nella variabile I il valore corrispondente al tasto premuto.
Linea 200: salta alla routine richiesta.
Linea 210: torna a mostrare le opzioni.

SEZIONE 3.
E' la sezione più complicata del programma: per comprenderne bene il funzionamento è preferibile suddividerla in 3 sottosezioni:
A) linee 1000-1050: disegna una griglia 8 per 8 su cui disegnare il carattere.
B) linee 1060-1170 e 1240: gestisce la posizione del cursore in seno alla griglia.
C) linee 1180-1230: riempie il vettore BY(7) con i dati relativi al carattere disegnato sulla griglia.

Torniamo al commento linea per linea.
Linea 1000: pulisce lo schermo, scrive in alto PREMI F1 PER USCIRE e pone il cursore di colore blu.
Linea 1010: sposta il cursore più in basso di 8 linee.
Linea 1020: per 8 volte esegue fino alla linea 1050.
Linea 1030: sposta il cursore a destra di 16 posizioni.
Linea 1040: scrive 8 volte +.
Linea 1050: porta il cusore a nuova linea e chiude il loop.
Linea 1060:pone l'indirizzo del byte della RAM del colore, corrispondente al + in alto a sinistra, nella variabile PC (posizione cursore), pone a zero le variabili XC e YC (coordinate X e Y del cursore nella griglia).
Linea 1070: la posizione del cursore diventa bianca (colore 1).
Linea 1080: accetta dalla tastiera un carattere qualunque.
Linea 1090: A = codice ASCII del carattere ricevuto da tastiera e SP = 0 (SP=spostamento).
Linea 1100: se il carattere ricevuto da tastiera è CURSORE A DESTRA e il cursore non è all'estrema destra della griglia (XC=7) allora incrementa XC e PC e pone SP = 1.
Linea 1110: se il carattere ricevuto da tastiera è CURSORE A SINISTRA e il cursore non è all'estrema sinistra della griglia (XC=0) allora decrementa XC e PC e pone SP = -1.
Linea 1120: se il carattere ricevuto da tastiera è CURSORE IN BASSO e il cursore non è nell'ultima linea della griglia (YC=7) allora incrementa YC, somma 40 a PC e pone SP = 40 (40 è il numero di caratteri contenuti in una linea dello schermo).
Linea 1130: se il carattere ricevuto da tastiera è CURSORE IN ALTO e il cursore non è nella prima linea della griglia (YC=0) allora decrementa YC, sottrae 40 da PC e pone SP = -40.
Linea 1140: se il carattere ricevuto da tastiera è SPAZIO pone nella posizione della mappa video corrispondente alla posizione del cursore uno spazio in campo inverso in modo da ANNERIRE un punto della griglia (il numero 54272 è la differenza di indirizzi tra mappa del colore e mappa video).
Linea 1150: se il carattere ricevuto da tastiera è DELETE pone nella posizione della mappa video corrispondente alla posizione del cursore un + in modo da CANCELLARE un punto della griglia.
Linea 1160: se il carattere ricevuto da tastiera è F1 rimette il cursore al colore blu

chiaro, colora di blu la posizione del cursore della griglia (colore 6).
Linea 1165: se il carattere ricevuto è F1 salta alla sottosezione C.
Linea 1170: ogni altro carattere viene ignorato; il programma colora la vecchia posizione del cursore di blu (se lo spostamento è stato nullo colora la posizione corrente del colore) e salta alla linea 1070 dove colora la posizione corrente del cursore di bianco.
Linea 1180: per ogni riga della griglia esegue fino a 1220.
Linea 1190: pone a zero il corrispondente elemento del vettore.
Linea 1200: per ogni elemento di una riga esegue fino a 1220.
Linea 1210: somma alla variabile corrispodente alla riga interessata 2 elevato alla posizione dell'elemento di riga considerato (partendo da destra); se in tale posizione vi è uno spazio in campo inverso, altrimenti 0 (nota che se esegui l'istruzione PRINT A=B il risultato è -1 se A=B, 0 se A < > B).
Linea 1220: chiude il ciclo degli elementi e quello delle righe. A questo punto nel vettore BY sono contenuti gli 8 numeri necessari per programmare il carattere disegnato sulla griglia.
Linea 1230: fine della routine.

SEZIONE 4.
Linee 2000-2055: mostra le opzioni relative alla memorizzazione dei caratteri.
Linee 2060-2070: accetta dalla tastiera un numero tra 1 e 5 e lo pone nella variabile A.
Linee 2080-2090: accetta un numero tra 0 e 511 e lo pone nella variabile NC% (il carattere che si vuole programmare).
Linea 2100: salta alla routine di modifica del vettore richiesta.
Linea 2110: memorizza il carattere ponendo in 8 byte di memoria a partire da 12288 + NC% * 8 il contenuto del vettore.
Linea 2120: torna dalla routine.

SEZIONE 5.
Linea 2200: è stata messa per rispondere alla chiamata della linea 2100.

SEZIONE 6.
Linea 2300: pone in ogni elemento del vettore (255-il valore già contenuto); questa operazione, compiuta su numeri minori di 256 (come nel nostro caso), nega gli 8 bit del numero.
Linea 2310: torna dalla routine.

SEZIONE 7.
Linea 2400: per ogni elemento del vettore esegue fino a 2450.
Linea 2410: per J da 0 a 3 esegue fino a 2450.
Linea 2420: pone il valore del bit j-esimo dell'elemento considerato del vettore nella variabile BD (bit di destra) e lo azzera.

Linea 2430-2435: pone il valore del bit (7-j)-esimo dell'elemento considerato del vettore nella variabile BS (bit di sinistra) e lo azzera.
Linea 2440: pone il bit di sinistra al posto del bit di destra e viceversa.
Linea 2450: chiude i loop di J e degli elementi del vettore.
Linea 2460: torna dalla routine.

SEZIONE 8.
Linea 2500: per i primi 4 elementi del vettore esegue fino a 2540.
Linea 2510: pone nella variabile T (temporanea) il valore dell'elemento considerato.
Linea 2520: pone nell'elemento considerato il valore dell'elemento simmetrico del vettore.
Linea 2530: pone nell'elemento simmetrico il valore di T.
Linea 2540: chiude il loop.
Linea 2550 torna dalla routine.

SEZIONE 9.
Linee 2600-2620: trasferisce il vettore BY nel vettore B2 e lo azzera.
Linea 2630: per ogni bit degli elementi del vettore esegue fino a 2660 (indice I).
Linea 2640: per ogni elemento del vettore esegue fino a 2660 (indice J).
Linea 2645-2650: pone il bit considerato dell'elemento considerato uguale al bit j-esimo dell'elemento simmetrico all'i-esimo.
Linea 2660: chiude i due loop.
Linea 2670: torna dalla routine.

SEZIONE 10.
Linee 3000-3020: chiede il numero (D-CODE) del carattere da correggere, accetta un numero tra 0 e 255 e lo pone nella variabile NC%.
Linea 3030: pulisce lo schermo, scrive centrato in alto PREMI F1 PER USCIRE e pone il cursore al colore blu.
Linea 3040: sposta il cursore più in basso di 8 linee.
Linea 3050: per 8 volte esegue fino alla linea 3100.
Linea 3060: sposta il cursore a destra di 16 posizioni.
Linee 3070-3100: scrive + se il bit corrispondente del carattere considerato contiene 0, uno spazio in campo inverso se contiene 1.
Linea 3110: salta alla routine di programmazione del carattere.

SEZIONE 11.
Linea 4000: pulisce lo schermo e scrive PREMI F1 PER TORNARE.
Linee 4005-4006: fornisce istruzioni all'utente.
Linea 4010: attende circa due secondi.
Linea 4020: cancella la scritta.
Linea 4030 pone la mappa dei caratteri in 12288 (3000H).
Linea 4040: accetta un carattere dalla tastiera.

Linea 4045-4055: se il tasto premuto è F1 riassume la ROM come mappa dei caratteri, pone il cursore al blu chiaro (è infatti possibile vedere i caratteri colorati come si preferisce) e torna dalla routine.
Linea 4060: scrive il carattere ricevuto dalla tastiera.
Linea 4070: torna a ricevere il prossimo carattere.

SEZIONE 12:
Linea 5000: pulisce lo schermo.
Linea 5010: chiede se si vuole salvare su disco o su nastro.
Linea 5015: accetta il carattere D o N.
Linea 5020: porta il cursore più giù di una linea.
Linea 5030: chiede il numero di set che si vuole salvare, NS%, per poter far stare su disco più di un set.
Linea 5040:porta il cursore più giù di una linea.
Linee 5050-5055: chiede il numero di caratteri che si vogliono salvare.
Linea 5060: se la risposta è maggiore di 256 caratteri (un set completo) torna a porre la domanda.
Linea 5070: se si vuole salvare su nastro salta alla linea 5500.
Linee 5080-5082: apre il file su disco ed esegue la routine di errore.
Linee 5085-5087: se si verifica errore attende circa 2 secondi per far leggere la scritta, chiude il file e il canale di comando e torna dalla routine.
Linee 5090-5110: scrive su disco il numero di caratteri da salvare e i caratteri.
Linea 5120: chiude il file e il canale di comando.
Linea 5130: torna dalla routine.
Linea 5500: apre il file su nastro.
Linee 5510-5530: scrive su nastro il numero di caratteri da salvare e i caratteri.
Linea 5540: chiude il file su nastro.
Linea 5550 torna dalla routine.

SEZIONE 13.
Linea 6000: pulisce lo schermo.
Linea 6010: chiede se si vuole caricare da disco o da nastro.
Linea 6015: accetta il carattere D o N.
Linea 6020: porta il cursore più giù di una linea.
Linea 6030: chiede il numero del set che si vuole caricare.
Linea 6070: se si vuole caricare da nastro salta alla linea 6500.
Linee 6080-6082: apre il file su disco ed esegue la routine di errore.
Linee 6085-6087: se si verifica errore attende circa 2 secondi per far leggere la scritta, chiude il file e il canale di comando e torna dalla routine.
Linee 6090-6110 legge dal disco il numero di caratteri da caricare e i caratteri.
Linea 6120: chiude il file e il canale di comando.
Linea 6130: torna dalla routine.
Linea 6500: apre il file su nastro.

Linee 6510-6530: carica da nastro il numero di caratteri da caricare e i caratteri.
Linea 6540: chiude il file su nastro.
Linea 6550: torna dalla routine.

SEZIONE 14.
Linea 7000: porta il cursore più giù di una linea e ti chiede se sei sicuro di voler abbandonare il programma.
Linea 7010:accetta un carattere dalla tastiera.
Linea 7020: se la risposta è S allora salta alla routine di inizializzazione del BASIC.
Linea 7030: altrimenti torna dalla routine.

SEZIONE 15.
Linee 9000-9005: abbassa il cursore di una linea, apre il canale di comando, riceve dal disco il messaggio e lo scrive sul video.
Linea 9010: torna dalla routine.

Con questo programma è quindi possibile creare in memoria nuovi caratteri e salvarli su disco o cassetta. Se, in un programma che usa caratteri definiti, vuoi evitare di caricare i dati relativi ai caratteri da cassetta, e preferisci che siano già nel programma sotto forma di linee DATA, puoi usare il programma GRAF6, che converte i byte contenuti dalla cella 12288 (3000H) in poi in linee DATA.

```
 10 REM GRAF6
 20 INPUT"FINO A CHE CARATTERE ";C%
 30 PRINT:I=0
 40 IFC%>255THENPRINT"MAX 255":PRINT:GOTO20
 50 PRINTCHR$(147);
 60 N=1000+I*10:GOSUB160:PRINTN$"DATA";
 70 FORJ=0TO7
 80 N=PEEK(12288+I*8+J):GOSUB160:PRINTN$",";
 90 NEXT
 100 PRINTCHR$(20)
 110 PRINT"I="I+1":C%="C%;
 120 IFI<C%THENPRINT":GOTO50":GOTO140
 130 PRINT":GOTO170"
 140 POKE198,3:POKE631,19:POKE632,13
 150 POKE633,13:END
 160 N$=STR$(N):N$=RIGHT$(N$,LEN(N$)-1):RETURN
 170 POKE198,3:POKE631,19:POKE632,13:POKE633,13
 180 K=K+10:PRINTCHR$(147)K
 190 PRINT"K="K":IFK<170THEN170"
```

GRAF6 che può essere usato anche per convertire in linee DATA programmi scritti in linguaggio macchina o altri tipi di dati presenti in memoria, si basa sul fatto che il COMMODORE 64, quando esce da un programma, scrive i dati che trova nel buffer della tastiera (631-640 (277H-280H)) e si comporta esattamente come se fossero stati premuti dall'utente. Osserva bene come funziona questo piccolo TRUCCO che verrà ripetuto anche in altri programmi di questo libro (vedi anche Paragrafo 4.6 del volume sul BASIC).

COMMENTO A GRAF6.

Linea 10: pone i puntatori di fine memoria a 12288 (3000H) per non cancellare, con le variabili, i dati posti da quella locazione in poi.
Linea 20: chiede fino a che carattere vuoi convertire in linea DATA. Il programma convertirà 8*(NC+1) byte a partire dall'indirizzo 12288.
Linee 30-40: scende di una linea e, se il numero di caratteri supera il massimo consentito per un set (256) torna alla linea 20. Questa linea può essere tolta quando si vuole convertire più dati.
Linea 50: pulisce lo schermo.
Linea 60: pone il numero che avrà la prossima linea DATA uguale a 1000+I*10, trasforma N in una stringa (appoggiandosi a una routine in 140), scrive la stringa contenente N e la parola BASIC DATA. Si può scegliere da quale linea partire con le linee DATA (sostituendo a 1000 il numero di linea desiderato) e con che passo incrementare il numero di linea (sostituendo a 10 il passo desiderato).
Linee 70-90: per gli 8 byte di ogni carattere pone in N il contenuto del byte interessato, lo trasforma in una stringa usando la stessa routine usata prima, scrive la stringa e una virgola.
Linea 100: cancella l'ultima virgola.
Linee 110-120: scrive sul video dei comandi BASIC che incrementeranno l'indice I e rimetteranno in memoria il valore di C (le variabili vengono infatti perse ogni volta che si introduce una nuova linea del programma). Se non è stato considerato l'ultimo carattere la linea 120 scrive anche GOTO 50.
Linea 130: se è stato considerato l'ultimo carattere viene scritto GOTO 170.
Linee 140-150: pone 3 nel byte 198 (il byte 198 indica il numero di caratteri validi del buffer di tastiera), pone nel buffer di tastiera il codice ASCII di HOME e due volte il codice ASCII di RETURN e esce dal programma. A questo punto il calcolatore trova nel buffer di tastiera HOME e quindi porta il cursore sulla linea DATA, un RETURN, e quindi introduce nel programma la linea DATA portando il cursore sulla seconda linea scritta dal programma; trova il secondo RETURN ed esegue quindi i comandi scritti sulla linea, aggiorna cioè le variabili e salta a 50 o a 170.
Linea 160: è la routine che trasforma il numero N nella corrispondente stringa N$ usando la funzione STR$ e scartando il carattere che contiene il segno.

Linee 170-190: usando lo stesso metodo descritto sopra cancella le linee da 10 a 170 compresa. Restano nel programma le linee 180 e 190; per evitare questo, puoi trasformare le 3 linee, da 170 a 190, in un unica linea con i separatori due punti (noi non l'abbiamo potuto fare per esigenze tipografiche).

2.4.3 Caratteri a sfondo programmabile

Come abbiamo già visto nel capitolo precedente, quando scrivi un carattere puoi sceglierne il colore, selezionando il colore del cursore o scrivendo nella memoria del colore il codice desiderato. In questo modo selezioni il colore dell'INCHIOSTRO con cui il COMMODORE 64 scrive il carattere, ma quello della CARTA (lo sfondo) rimane sempre lo stesso. Il VIC II ha la capacità di poter leggere diversi colori di sfondo per ogni carattere quando viene posto in modo SFONDO PROGRAMMABILE. In questo modo, infatti, quando il VIC II legge dalla mappa video il codice di un carattere considera i 6 bit meno significativi come codice del carattere, e i due bit rimanenti come codice del colore dello sfondo. Hai così a disposizione un set di soli 64 (2 elevato 6) caratteri, ma puoi scegliere per ciascuno di essi un colore di sfondo su 4 possibili. Puoi naturalmente scegliere a tuo piacere questi 4 colori tra i 16 a disposizione. Con il modo a sfondo programmabile potrai fare delle bellissime maschere o dei coloratissimi quadri grafici. I caratteri che si perdono con il modo a sfondo programmabile sono i caratteri in campo inverso in entrambi i set, i caratteri grafici nel set grafico, le maiuscole e i caratteri grafici nel set commerciale. I 4 colori che vengono scelti come possibili sfondi vanno posti nel registro 33 (21H), lo stesso registro del colore dello schermo, nel registro 34 (22H), nel registro 35 (23H) e nel registro 36 (24H) che rispondono agli indirizzi 53281, 53282, 53283, 53284.

- Il primo colore viene dato come sfondo ai caratteri che hanno nei due bit più significativi 00.
- Il secondo colore viene dato come sfondo ai caratteri che hanno nei due bit più significativi 01.
- Il terzo colore viene dato come sfondo ai caratteri che hanno nei due bit più significativi 10.
- Il quarto colore viene dato come sfondo ai caratteri che hanno nei due bit più significativi 11.

Ricorda che il bit più significativo è a 1 per i caratteri in campo inverso. Nel modo a sfondo programmabile premendo A si ottiene sul video una normale A, premendo SHIFT-A si ottiene una A su un campo di colore 2, una A in campo inverso diventa una A su campo di colore 3, e uno SHIFT-A in campo inverso diventa una A su campo di colore 4. Questa regola vale per tutte le lettere dell' alfabeto, la freccia in alto (elevato a) e lo spazio. Non vale, purtroppo per i numeri e i segni di punteggiatura; per questi caratteri riportiamo la Tabella 2.4 per trovare le corrispondenze. Per entrare in modo a sfondo programmabile basta porre a 1 il bit di posizione 6 del

registro 17 del VIC II che risponde all'indirizzo 53265 (D011H). L'istruzione è la seguente:
POKE 53265,PEEK(53265) OR 64
per uscire dal modo a sfondo programmabile bisogna scrivere:
POKE 53265,PEEK(53265) AND 191.
Come prova di programmazione dello sfondo dei caratteri, segue il programma GRAF7.

```
1 REM GRAF7
10 PRINTCHR$(147):FORI=1TO8:PRINTCHR$(17):NEXT
20 POKE53265,PEEK(53265)OR64
30 POKE53280,0
40 POKE53281,0
50 POKE53282,12
60 POKE53283,14
70 POKE53284,2
80 PRINTCHR$(147);:FORI=1TO5:PRINT:NEXT
90 PRINTTAB(13)CHR$(30)"COMMODORE 64"
100 PRINT:PRINT
101 PRINTTAB(9)CHR$(28)"                      "
103 PRINTTAB(9)" ┐└┌┐│─♣╲╲♣ ─╮ ┐└┌╳♣ "
105 PRINTTAB(9)"                      "
110 PRINT
111 PRINTTAB(13)CHR$(31)CHR$(18)"             "
113 PRINTTAB(13)CHR$(18)" CARATTERI "
115 PRINTTAB(13)CHR$(18)"           "
120 PRINT:PRINTTAB(7)CHR$(158)CHR$(18);
121 PRINT"                          "
123 PRINTTAB(7)CHR$(18)" ♣ ♣─┌╱┬ ┐└┌┐│─♣╲╲♣│╮└▔ "
125 PRINTTAB(7)CHR$(18);
127 PRINT"                          "
130 GETA$:IFA$=""THEN130
140 POKE53265,PEEK(53265)AND191
```

COMMENTO A GRAF7.
Linea 20: entra nel modo a sfondo programmabile ponendo a 1 il bit di posizione 6 del registro 17 del VIC II.
Linea 30: colore del bordo = nero.
Linea 40: colore dello schermo (primo colore di sfondo) = nero.

Linea 50: secondo colore di sfondo = grigio.
Linea 60: terzo colore di sfondo = blu chiaro.
Linea 70: quarto colore di sfondo = rosso.
Linea 80: pulisce lo schermo e porta il cursore sulla quinta linea.
Linea 90: scrive, al centro, in verde: COMMODORRE 64.
Linee 100-101: scrive 3 righe più in basso, al centro, 20 spazi SHIFTATI e cambia il colore del cursore in rosso.
Linea 103: scrive, al centro la scritta SHIFTATA: PROGRAMMA DI PROVA. Anche gli spazi sono SHIFTATI.
Linea 105: scrive, al centro, 20 spazi SHIFTATI.
Linee 110-111: scrive due righe più in basso al centro, 11 spazi IN CAMPO INVERSO e cambia il colore del cursore in blu.
Linea 113: scrive, al centro la scritta IN CAMPO INVERSO: CARATTERI.
Linea 115: scrive, al centro, 11 spazi IN CAMPO INVERSO.
Linee 120-121: scrive due righe più in basso al centro, 24 spazi IN CAMPO INVERSO e SHIFTATI e cambia il colore del cursore in blu chiaro.
Linea 123: scrive, al centro la scritta IN CAMPO INVERSO e SHIFTATA: A SFONDO PROGRAMMABILE. Anche gli spazi sono SHIFTATI.
Linea 125: scrive, al centro, 24 spazi IN CAMPO INVERSO e SHIFTATI.

Ti proponiamo ora il programma GRAF8,leggermente più complicato che mostra il modo a sfondo programmabile applicato ai caratteri programmabili: in questo esempio DISEGNAMO alcuni caratteri e li disponiamo sul video più volte con sfondi diversi:

```
1 REM GRAF8
10 POKE56,48:POKE55,0:CLR:DM=55296-1024
20 FORI=12288TO12479
30 READA
40 POKEI,A
50 NEXT
55 FORI=0TO7:POKE12544+I,0:NEXT
60 PRINTCHR$(147)
70 POKE53280,1:POKE53281,14
80 POKE53272,(PEEK(53272)AND240)+12
90 POKE53265,PEEK(53265)OR64
100 POKE53282,5
110 POKE53283,8
120 POKE53284,7
130 FORI=1544TO1703:POKEI,96:NEXT
140 FORI=1704TO1863:POKEI,160:NEXT
```

```
150 FORI=1864TO2023:POKEI,96:NEXT
160 FORI=0TO2:FORJ=0TO2:POKE1501+J+I*40,224
165 NEXT:NEXT
170 PRINTCHR$(19);:FORI=0TO7:PRINTCHR$(17);:NEXT
175 PRINTCHR$(157)CHR$(28)"V";
180 A$="VWWW":FORI=0TO2
190 FORJ=0TO37-I:PRINTCHR$(29);:NEXT
200 PRINTLEFT$(A$,I+2);
210 NEXT
220 OG=1540:T=1:GOSUB300
240 OG=1630:T=0:GOSUB300
250 OG=1490:T=0:GOSUB300
260 OG=1800:T=1:GOSUB300
270 GOTO270
300 IFTTHEN420
310 POKEOG,PEEK(OG)-32:POKEOG+DM,1:OG=OG+1
320 POKEOG,PEEK(OG)-32+1:POKEOG+DM,1:OG=OG+39
330 POKEOG,PEEK(OG)-32+2:POKEOG+DM,6:OG=OG+1
340 POKEOG,PEEK(OG)-32+3:POKEOG+DM,6:OG=OG+38
350 POKEOG,PEEK(OG)-32+6:POKEOG+DM,6:OG=OG+1
360 POKEOG,PEEK(OG)-32+4:POKEOG+DM,6:OG=OG+1
370 POKEOG,PEEK(OG)-32+5:POKEOG+DM,6:OG=OG+39
380 POKEOG,PEEK(OG)-32+7:POKEOG+DM,1:OG=OG+1
390 POKEOG,PEEK(OG)-32+8:POKEOG+DM,1:OG=OG+39
400 POKEOG,PEEK(OG)-32+9:POKEOG+DM,1:OG=OG+1
410 POKEOG,PEEK(OG)-32+10:POKEOG+DM,1:RETURN
420 POKEOG,PEEK(OG)-32+12:POKEOG+DM,1:OG=OG+1
430 POKEOG,PEEK(OG)-32+11:POKEOG+DM,1:OG=OG+39
440 POKEOG,PEEK(OG)-32+14:POKEOG+DM,6:OG=OG+1
450 POKEOG,PEEK(OG)-32+13:POKEOG+DM,6:OG=OG+39
460 POKEOG,PEEK(OG)-32+16:POKEOG+DM,6:OG=OG+1
470 POKEOG,PEEK(OG)-32+15:POKEOG+DM,6:OG=OG+1
480 POKEOG,PEEK(OG)-32+17:POKEOG+DM,6:OG=OG+38
490 POKEOG,PEEK(OG)-32+19:POKEOG+DM,1:OG=OG+1
500 POKEOG,PEEK(OG)-32+18:POKEOG+DM,1:OG=OG+39
510 POKEOG,PEEK(OG)-32+21:POKEOG+DM,1:OG=OG+1
520 POKEOG,PEEK(OG)-32+20:POKEOG+DM,1:RETURN
1000 DATA0,0,1,3,7,15,31,31
1010 DATA0,96,240,240,224,192,192,224
1020 DATA31,63,63,127,127,63,31,7
1030 DATA224,54,191,255,246,240,224,0
1040 DATA7,31,62,125,251,247,207,159
```

```
1050 DATA0,176,216,220,238,239,231,195
1060 DATA0,0,0,0,1,7,7,15
1070 DATA31,31,15,23,27,29,30,60
1080 DATA192,128,128,225,243,119,127,62
1090 DATA126,255,126,0,0,0,0,0
1100 DATA28,0,0,0,0,0,0,0
1110 DATA0,0,128,192,224,240,248,248
1120 DATA0,6,15,15,7,3,3,7
1130 DATA248,252,252,254,254,252,248,224
1140 DATA7,108,253,255,111,15,7,0
1150 DATA224,248,124,190,223,239,243,249
1160 DATA0,13,27,59,119,247,231,195
1170 DATA0,0,0,0,128,224,224,240
1180 DATA248,248,240,232,216,184,120,60
1190 DATA3,1,1,135,207,238,254,124
1200 DATA126,255,126,0,0,0,0,0
1210 DATA56,0,0,0,0,0,0,0
1220 DATA1,3,7,15,31,63,127,255
1230 DATA255,255,255,255,255,255,255,255
```

GRAF8 crea 22 caratteri necessari a disegnare dei puffi e, grazie a una subroutine, li pone in un qualsiasi punto di uno schermo variopinto.

COMMENTO A GRAF8.
Linea 10: pone la fine della memoria a 12288 (3000H) per proteggere i caratteri dalle variabili e pone nella variabile DM (differenza mappe) la differenza tra il primo indirizzo della mappa del colore e il primo indirizzo della mappa video.
Linee 20-50: trasferisce i dati delle immagini in memoria.
Linea 55: pone 8 zeri nel carattere 32 (lo spazio).
Linea 60: pulisce lo schermo.
Linea 70: bordo nero e schermo (colore sfondo 1) azzurro.
Linea 80: avvisa il VIC II che la mappa dei caratteri è in 12288 (3000H).
Linea 90: entra in modo a sfondo programmabile.
Linea 100: colore sfondo 2 = verde.
Linea 110: colore sfondo 3 = giallo.
Linea 120: colore sfondo 4 = marrone.
Linea 130: disegna una striscia verde usando degli spazi SHIFTATI (codice schermo = 96).
Linea 140: disegna una striscia marrone usando degli spazi IN CAMPO INVERSO (codice schermo = 160).

Linea 150: disegna una seconda striscia verde usando spazi SHIFTATI.
Linea 160: disegna una casetta gialla usando spazi IN CAMPO INVERSO e SHIFTATI.
Linee 170-210: disegna il tetto della casa usando dei caratteri appositamente programmati. Nota che in questa zona del video non potrai mettere dei puffi perchè il tetto non è stato ottenuto con degli spazi a sfondo colorato come nel caso delle strisce colorate o della casetta.
Linea 220: disegna un puffo di tipo 1 (che guarda a sinistra), il cui angolo in alto a sinistra verrà posto nel byte della memoria video 1540 usando la subroutine in 300.
Linea 240: disegna un puffo di tipo 0 (che guarda a destra), il cui angolo in alto a sinistra verrà posto nel byte della memoria video 1630 usando la subroutine in 300.
Linea 250: disegna un puffo di tipo 0 (che guarda a destra), il cui angolo in alto a sinistra verrà posto nel byte della memoria video 1490 usando la subroutine in 300.
Linea 260: disegna un puffo di tipo 1 (che guarda a sinistra), il cui angolo in alto a sinistra verrà posto nel byte della memoria video 1800 usando la subroutine in 300.
Linea 270: ferma il programma (per uscirne dovrai premere STOP/RESTORE)
Linea 300: salta se T < > 0 quindi se il puffo è di tipo 1.
Linee 310-410: ogni linea disegna uno degli 11 caratteri che compongono un puffo di tipo 0 e pone nel corrispondente byte del colore il colore adatto. Per disegnare un carattere in un byte viene sottratto dal contenuto del byte il numero 32 (codice dello spazio) e sommato il codice desiderato: in questo modo i due bit più significativi rimangono inalterati e quindi non cambia il colore dello sfondo.
Linee 420-520: ogni linea disegna uno degli 11 caratteri che compongono un puffo di tipo 1 e pone nel corrispondente byte del colore il colore adatto.
Linee 1000-1230: dati relativi ai caratteri dei puffi e del tetto della casetta.

TABELLA DI CORRISPONDENZA				
SIMB.	TASTI DA PREMERE			
	COLORE 1	COLORE 2	COLORE 3	COLORE 4
@	@	SHIFT+*	RVS+@	RVS+SHIFT+*
A	A	SHIFT+A	RVS+A	RVS+SHIFT+A
B	B	SHIFT+B	RVS+B	RVS+SHIFT+B
C	C	SHIFT+C	RVS+C	RVS+SHIFT+C
D	D	SHIFT+D	RVS+D	RVS+SHIFT+D
E	E	SHIFT+E	RVS+E	RVS+SHIFT+E
F	F	SHIFT+F	RVS+F	RVS+SHIFT+F
G	G	SHIFT+G	RVS+G	RVS+SHIFT+G
H	H	SHIFT+H	RVS+H	RVS+SHIFT+H
I	I	SHIFT+I	RVS+I	RVS+SHIFT+I
J	J	SHIFT+J	RVS+J	RVS+SHIFT+J
K	K	SHIFT+K	RVS+K	RVS+SHIFT+K
L	L	SHIFT+L	RVS+L	RVS+SHIFT+L
M	M	SHIFT+M	RVS+M	RVS+SHIFT+M
N	N	SHIFT+N	RVS+N	RVS+SHIFT+N
O	O	SHIFT+O	RVS+O	RVS+SHIFT+O

Tabella 2.4 Corrispondenze TASTI-CODICE COLORE per caratteri a sfondo programmabile

TABELLA DI CORRISPONDENZA				
SIMB.	TASTI DA PREMERE			
	COLORE 1	COLORE 2	COLORE 3	COLORE 4
P	P	SHIFT+P	RVS+P	RVS+SHIFT+P
Q	Q	SHIFT+Q	RVS+Q	RVS+SHIFT+Q
R	R	SHIFT+R	RVS+R	RVS+SHIFT+R
S	S	SHIFT+S	RVS+S	RVS+SHIFT+S
T	T	SHIFT+T	RVS+T	RVS+SHIFT+T
U	U	SHIFT+U	RVS+U	RVS+SHIFT+U
V	V	SHIFT+V	RVS+V	RVS+SHIFT+V
W	W	SHIFT+W	RVS+W	RVS+SHIFT+W
X	X	SHIFT+X	RVS+X	RVS+SHIFT+X
Y	Y	SHIFT+Y	RVS+Y	RVS+SHIFT+Y
Z	Z	SHIFT+Z	RVS+Z	RVS+SHIFT+Z
[	[	SHIFT++	RVS+[	RVS+SHIFT++
£	£	CBM+-	RVS+£	RVS+CBM+-
]	]	SHIFT+-	RVS+]	RVS+SHIFT+-
↑	↑	SHIFT+↑	RVS+↑	RVS+SHIFT+↑
←	←	CBM+*	RVS+←	RVS+CBM+*

Tabella 2.4 Corrispondenze TASTI-CODICE COLORE per caratteri a sfondo programmabile (continuazione)

TABELLA DI CORRISPONDENZA				
SIMB.	TASTI DA PREMERE			
	COLORE 1	COLORE 2	COLORE 3	COLORE 4
SP	SP	SHIFT+SP	RVS+SP	RVS+SHIFT+SP
!	!	CBM+K	RVS+!	RVS+CBM+K
"	"	CBM+I	RVS+"	RVS+CBM+I
#	#	CBM+T	RVS+#	RVS+CBM+T
$	$	CBM+@	RVS+$	RVS+CBM+@
%	%	CBM+G	RVS+%	RVS+CBM+G
&	&	CBM++	RVS+&	RVS+CBM++
'	'	CBM+M	RVS+'	RVS+CBM+M
(	(	CBM+£	RVS+(	RVS+CBM+£
)	)	SHIFT+£	RVS+)	RVS+SHIFT+£
*	*	CBM+N	RVS+*	RVS+CBM+N
+	+	CBM+Q	RVS++	RVS+CBM+Q
,	,	CBM+D	RVS+,	RVS+CBM+D
-	-	CBM+Z	RVS+-	RVS+CBM+Z
.	.	CBM+S	RVS+.	RVS+CBM+S
/	/	CBM+P	RVS+/	RVS+CBM+P

Tabella 2.4 Corrispondenze TASTI-CODICE COLORE per caratteri a sfondo programmabile (continuazione)

TABELLA DI CORRISPONDENZA				
SIMB.	TASTI DA PREMERE			
	COLORE 1	COLORE 2	COLORE 3	COLORE 4
0	0	CBM+A	RVS+0	RVS+CBM+A
1	1	CBM+E	RVS+1	RVS+CBM+E
2	2	CBM+R	RVS+2	RVS+CBM+R
3	3	CBM+W	RVS+3	RVS+CBM+W
4	4	CBM+H	RVS+4	RVS+CBM+H
5	5	CBM+J	RVS+5	RVS+CBM+J
6	6	CBM+L	RVS+6	RVS+CBM+L
7	7	CBM+Y	RVS+7	RVS+CBM+Y
8	8	CBM+U	RVS+8	RVS+CBM+U
9	9	CBM+O	RVS+9	RVS+CBM+O
:	:	SHIFT+@	RVS+:	RVS+SHIFT+@
;	;	CBM+F	RVS+;	RVS+CBM+F
<	<	CBM+C	RVS+<	RVS+CBM+C
=	=	CBM+X	RVS+=	RVS+CBM+X
>	>	CBM+V	RVS+>	RVS+CBM+V
?	?	CBM+B	RVS+?	RVS+CBM+B

Tabella 2.4 Corrispondenze TASTI-CODICE COLORE per caratteri a sfondo programmabile (continuazione)

2.4.4 Caratteri multicolore

Questo paragrafo tratta dell'ultimo modo che hai a disposizione per rappresentare i caratteri con il COMMODORE 64: il modo multicolore. Nel modo alta risoluzione (il modo normale) puoi colorare ogni puntino che forma un carattere agendo sul bit ad esso corrispondente. Ogni punto può valere 0 o 1, cioè essere colorato o no. Nel modo multicolore ad ogni punto che forma il carattere corrispondono 2 bit: in questo modo ogni punto può essere di un colore scelto tra 4 (i colori di sfondo 1, 2, 3 e il colore proprio della locazione video). L'unica cosa che viene persa, passando dal modo ad alta risoluzione al modo multicolore, è la risoluzione orizzontale: i caratteri non corrispondono più a matrici 8*8, ma corrispondono a matrici 4*8.
Per entrare in modo multicolore bisogna porre a 1 il bit di posizione 4 del registro 22 del VIC II (indirizzo 53270 (D016H)): bisogna perciò eseguire il seguente comando:
POKE 53270,PEEK(53270) OR 16.
Per tornare in modo alta risoluzione dovrai eseguire il comando:
POKE53270,PEEK(53270) AND 239.
Per chiarire le idee vediamo come il VIC II legge il carattere A in modo multicolore:
Ecco il contenuto degli 8 byte relativi alla A:

```
0 0 0 1 1 0 0 0
0 0 1 1 1 1 0 0
0 1 1 0 0 1 1 0
0 1 1 1 1 1 1 0
0 1 1 0 0 1 1 0
0 1 1 0 0 1 1 0
0 1 1 0 0 1 1 0
0 0 0 0 0 0 0 0
```

Il VIC II, nel modo multicolore li legge, come 4 punti colore per riga, così:

```
00 01 10 00     cioè come:     A B C A
00 11 11 00                    A D D A
01 10 01 10                    B C B C
01 11 11 10                    B D D C
01 10 01 10                    B C B C
01 10 01 10                    B C B C
01 10 01 10                    B C B C
00 00 00 00                    A A A A
```

dove A (da non confondere con il carattere A preso come spunto), B, C e D sono dei punti colore e:
A è del colore dello schermo
B è del colore di sfondo 2 (vedi 2.4.3)
C è del colore di sfondo 3
D è del colore proprio del byte.

I primi 3 colori possono essere scelti tra i 16 del COMMODORE 64, mentre il quarto può essere solo uno dei primi 8. Questa piccola limitazione permette, in compenso, di visualizzare contemporaneamente caratteri in multicolore e ad alta risoluzione, il VIC II infatti visualizza il carattere come multicolore solo se il byte corrispondente della mappa del colore ha il bit di posizione 3 a 1 e assume come codice del colore il numero formato dai 3 bit meno significativi.
Ecco GRAF9, come esempio di programmazione di un carattere multicolore:

```
1 REM GRAF9
10 PRINTCHR$(147)CHR$(150)
20 POKE53270,PEEK(53270)OR16
30 FORI=0TO7:READA:POKE12288+I,A:NEXT
40 FORI=0TO7:POKE12288+32*8+I,0:NEXT
50 POKE53280,0
60 POKE53281,0
70 POKE53282,5
80 POKE53283,6
90 POKE53272,(PEEK(53272)AND240)+12
100 PRINT"@"
110 GETA$:IFA$=""THEN110
120 POKE53270,PEEK(53270)AND239
130 POKE53272,(PEEK(53272)AND240)+5
140 PRINTCHR$(154)
1000 DATA165,165,165,165,240,240,240,240
```

COMMENTO A GRAF9.
Linea 10: pulisce lo schermo e pone il colore del cursore rosa (codice 10). Il colore di un carattere multicolore scritto con il codice 10 sarà quindi del colore 10-8=2 (rosso).
Linea 20: entra in modo multicolore.
Linea 30: legge gli 8 dati relativi al carattere chiocciola e li pone a partire da 12288 (3000H).

Linea 40: azzera gli 8 byte relativi al carattere di codice 32 (lo spazio).
Linea 50: colore del bordo = nero.
Linea 60: colore dello schermo = nero.
Linea 70: colore di sfondo 2 = verde.
Linea 80: colore di sfondo 3 = blu.
Linea 90: pone la mappa dei caratteri in 12288 (3000H).
Linea 100: stampa il carattere di D-CODE uguale a 0 (la chiocciola).
Linea 110: attende che sia premuto un tasto.
Linea 120: torna in modo alta risoluzione.
Linea 130: riattiva i caratteri dalla ROM.
Linea 140: colore del cursore blu chiaro.
Linea 1000: contiene i dati relativi al carattere.

Questo esempio mostra chiaramente come il VIC II sia capace di porre 4 colori diversi nella matrice di un solo carattere, ma il risultato non è certamente dei più pittoreschi. Prova quindi il programma GRAF10, il cui scopo è quello di disegnare dei coloratissimi omini spaziali.

```
1 REM GRAF10
10 PRINTCHR$(147)CHR$(155)
20 POKE53270,PEEK(53270)OR16
30 FORI=0TO47:READA:POKE12288+I,A:NEXT
40 FORI=0TO7:POKE12288+32*8+I,0:NEXT
50 POKE53280,0
60 POKE53281,0
70 POKE53282,2
80 POKE53283,6
90 POKE53272,(PEEK(53272)AND240)+12
100 FORI=0TO9:PRINT:NEXT
110 PRINT" @A @A @A @A @A @A @A";
115 PRINT" @A @A @A @A @A @A"
120 PRINT" BC BC BC BC BC BC BC";
125 PRINT" BC BC BC BC BC BC"
130 PRINT" DE DE DE DE DE DE DE";
135 PRINT" DE DE DE DE DE DE"
140 GETA$:IFA$=""THEN140
150 POKE53270,PEEK(53270)AND239
160 POKE53272,(PEEK(53272)AND240)+5
170 PRINTCHR$(154)
1000 DATA192,192,49,53,21,81,65,85
```

```
1010 DATA3,3,76,92,84,69,65,85
1020 DATA84,85,22,5,3,5,21,85
1030 DATA21,85,148,80,192,80,84,85
1040 DATA85,85,21,5,1,34,42,10
1050 DATA85,85,84,80,64,136,168,160
```

COMMENTO A GRAF10.
Linea 10: pulisce lo schermo e pone il colore del cursore a grigio chiaro (codice 15). Il colore di un carattere multicolore scritto col codice 15 sarà quindi di codice 15-8=7 (giallo).
Linea 20: entra in modo multicolore.
Linea 30: legge i 48 dati relativi ai 6 caratteri e li pone a partire da 12288 (3000H).
Linea 40: azzera gli 8 byte relativi al carattere di codice 32 (lo spazio).
Linea 50: colore del bordo = nero.
Linea 60: colore dello schermo = nero.
Linea 70: colore di sfondo 2 = rosso.
Linea 80: colore di sfondo 3 = blu.
Linea 90: pone la mappa dei caratteri in 12288 (3000H).
Linea 100: porta il cursore 10 linee più in basso.
Linea 110: disegna 13 teste.
Linea 120: disegna 13 corpi.
Linea 130: disegna 13 paia di piedi.
Linea 140: attende che sia premuto un tasto.
Linea 150: torna in modo alta risoluzione.
Linea 160: riattiva i caratteri della ROM.
Linea 170: colore del cursore blu chiaro.
Linea 1000: contiene i dati relativi al carattere.

Come per i caratteri creati nel modo alta risoluzione, è possibile scrivere un programma che aiuti nella creazione dei caratteri multicolore. Riportiamo il listato completo del programma GRAF11 che svolge questa funzione. Il commento è limitato a quelle parti in cui differisce dal programma GRAF5 per la creazione dei caratteri ad alta risoluzione.

```
1 REM GRAF11
10 POKE13*4096+32,0:POKE13*4096+33,0
15 PRINTCHR$(154)CHR$(142)CHR$(8)
20 POKE56,48:POKE55,0:CLR
25 US$=CHR$(147)+"PREMI F1 PER USCIRE"
```

```
30 DIMC(15):FORI=0TO15:READC(I):NEXT
40 GOSUB8000
100 PRINTCHR$(147)
110 PRINT"OPZIONI  :":PRINT:PRINT
120 PRINT"1 CREA CARATTERE":PRINT
130 PRINT"2 MEMORIZZA CARATTERE":PRINT
140 PRINT"3 CORREGGE CARATTERE":PRINT
150 PRINT"4 MOSTRA CARATTERI":PRINT
160 PRINT"5 SALVA SET DI CARATTERI":PRINT
170 PRINT"6 CARICA SET DI CARATTERI":PRINT
180 PRINT"7 SCELTA COLORI":PRINT
185 PRINT"8 FINE"
190 GETA$:IFVAL(A$)=0THEN190
195 I=VAL(A$)
197 IFI=8THENGOSUB7000
200 ONIGOSUB1000,2000,3000,4000,5000,6000,8000
210 GOTO100
1000 GOSUB1500
1005 PRINTUS$:FORI=1TO200:NEXT
1010 POKE53281,-11*(CL=0):PRINTCHR$(C(CL));
1015 FORI=1TO8:PRINTCHR$(17);:NEXTI
1020 FORI=1TO8
1030 FORJ=1TO16:PRINTCHR$(32);:NEXT
1040 PRINTCHR$(18);
1045 FORJ=1TO8:PRINTCHR$(32);:NEXTJ
1050 PRINT:NEXTI
1060 PC=1400:XC=0:YC=0
1070 POKEPC,86:POKEPC+1,86
1080 GETA$:IFA$=""THEN1080
1090 A=ASC(A$):SP=0
1100 IFA=29ANDXC<6THENXC=XC+2:PC=PC+2:SP=2
1110 IFA=157ANDXC>0THENXC=XC-2:PC=PC-2:SP=-2
1120 IFA=17ANDYC<7THENYC=YC+1:PC=PC+40:SP=40
1130 IFA=145ANDYC>0THENYC=YC-1:PC=PC-40:SP=-40
1140 IFA=134THENPOKEPC+54272,CC:POKEPC+54273,CC
1141 IFA=135THENPOKEPC+54272,PEEK(53282)
1142 IFA=135THENPOKEPC+54273,PEEK(53282)
1143 IFA=136THENPOKEPC+54272,PEEK(53283)
1144 IFA=136THENPOKEPC+54273,PEEK(53283)
1150 IFA=20THENPOKEPC+54272,CL:POKEPC+54273,CL
1160 IFA=133THENPRINTCHR$(154):POKEPC,160
1165 IFA=133THENPOKEPC+1,160:POKE53281,0
```

```
1167 IFA=133THENGOTO1180
1170 POKEPC-SP,160:POKEPC-SP+1,160:GOTO1070
1180 FORI=0TO7
1190 BY(I)=0
1200 FORJ=0TO6STEP2
1205 P1=PEEK(55672+40*I+6-J)AND15
1207 P2=(PEEK(53282)AND15)
1208 P3=(PEEK(53283)AND15)
1210 IFP1=CCTHENBY(I)=BY(I)+2↑J+2↑(J+1)
1211 IFP1=P2THENBY(I)=BY(I)+2↑J
1212 IFP1=P3THENBY(I)=BY(I)+2↑(J+1)
1220 NEXTJ:NEXTI
1230 RETURN
1500 PRINTCHR$(147)
1510 PRINT:PRINT:PRINT:PRINT
1520 FORI=0TO7
1530 PRINTTAB(14)CHR$(C(I))CHR$(18)"        ";
1535 PRINTCHR$(154)CHR$(146)" ="I
1540 NEXT
1550 PRINTCHR$(19)TAB(10)"COLORE DI CELLA ";
1555 INPUTCC
1560 RETURN
2000 PRINTCHR$(147)
2010 PRINT"1 NORMALE":PRINT
2030 PRINT"2 SIMMETRIA VERTICALE":PRINT
2040 PRINT"3 SIMMETRIA ORIZZONTALE":PRINT
2060 GETA$:IFA$=""THEN2060
2070 A=VAL(A$):IFA=0ORA>3THEN2060
2080 INPUT"CARATTERE NUMERO ";NC%
2090 IFNC%>511ORN%<0THEN2080
2100 ONAGOSUB2200,2400,2500
2110 FORI=0TO7:POKE12288+8*NC%+I,BY(I):NEXT
2120 RETURN
2200 RETURN
2400 FORI=0TO7
2410 FORJ=0TO2STEP2
2420 LR=(BY(I)AND2↑J)/2↑J:BY(I)=BY(I)-2↑J*LR
2425 MR=(BY(I)AND2↑(J+1))/2↑(J+1)
2427 BY(I)=BY(I)-2↑(J+1)*MR
2430 ML=(BY(I)AND2↑(7-J))/2↑(7-J)
2432 BY(I)=BY(I)-2↑(7-J)*ML
```

```
2433 LL=(BY(I)AND2↑(6-J))/2↑(6-J)
2434 BY(I)=BY(I)-2↑(6-J)*LL
2440 BY(I)=BY(I)+2↑J*LL+2↑(J+1)*ML
2445 BY(I)=BY(I)+2↑(7-J)*MR+2↑(6-J)*LR
2450 NEXTJ,I
2460 RETURN
2500 FORI=0TO3
2510 T=BY(I)
2520 BY(I)=BY(7-I)
2530 BY(7-I)=T
2540 NEXTI
2550 RETURN
3000 PRINTCHR$(147)
3010 INPUT"CARATTERE DA CORREGGERE ";NC%
3020 IFNC%>511ORN%<0THEN3010
3030 GOSUB1500
3040 PRINTUS$
3050 FORI=0TO7:FORJ=0TO6STEP2
3055 POKE1400+I*40+J,160:POKE1401+I*40+J,160
3060 FORJ=0TO6STEP2
3070 POKE1400+I*40+J,160
3080 POKE1401+I*40+J,160
3090 A=PEEK(12288+8*NC%+I)AND(2↑(7-J)+2↑(6-J))
3100 IFA=0THENPOKE55672+I*40+J,CL
3110 IFA=0THENPOKE55673+I*40+J,CL
3115 T7=A=2↑(7-J):T6=A=2↑(6-J)
3120 IFT6THENPOKE55672+40*I+J,PEEK(53282)AND15
3130 IFT6THENPOKE55673+40*I+J,PEEK(53282)AND15
3140 IFT7THENPOKE55672+40*I+J,PEEK(53283)AND15
3150 IFT7THENPOKE55673+40*I+J,PEEK(53283)AND15
3160 IFA=3*2↑(6-J)THENPOKE55672+40*I+J,CC
3165 IFA=3*2↑(6-J)THENPOKE55673+40*I+J,CC
3170 NEXT:NEXT
3180 GOTO1060
4000 PRINTUS$
4005 PRINTCHR$(17)"PREMI I TASTI CORRISPONDENTI"
4006 PRINT"AI CARATTERI CHE VUOI VEDERE"
4010 FORI=1TO2000:NEXTI
4020 PRINTCHR$(147);:POKE53281,CL
4030 POKE53272,(PEEK(53272)AND240)+12
4035 POKE53270,PEEK(53270)OR16
```

```
4040 GETA$:IFA$=""THEN4040
4045 F1=A$=CHR$(133)
4050 IFF1THENPOKE53272,(PEEK(53272)AND240)+5
4052 IFF1THENPRINTCHR$(154)
4055 IFF1THENPOKE53281,0
4057 IFF1THENPOKE53270,PEEK(53270)AND239:RETURN
4060 PRINTA$;
4070 GOTO4040
5000 PRINTCHR$(147)
5010 PRINT"DISCO O NASTRO ?"
5015 GETDV$:IFDV$<>"N"ANDDV$<>"D"THEN5015
5020 PRINT
5030 INPUT"SET NUMERO ";NS%
5040 PRINT
5050 PRINT"FINO A CHE CARATTERE VUOI SALVARE "
5055 INPUTNC%
5060 IFNC%>255THEN5040
5070 IFDV$="N"THEN5500
5080 OPEN1,8,2,"@:SET #"+STR$(NS%)+",S,W"
5082 GOSUB9000
5085 IFNNTHENFORI=1TO2000:NEXT:CLOSE1:CLOSE15
5087 IFNNTHENRETURN
5090 PRINT#1,CHR$(NC%);:FORI=0TO(NC%+1)*8-1
5100 PRINT#1,CHR$(PEEK(12288+I));
5110 NEXTI
5120 CLOSE1:CLOSE15
5130 RETURN
5500 OPEN1,1,1,"SET #"+STR$(NS%)
5510 PRINT#1,CHR$(NC%);:FORI=0TO(NC%+1)*8-1
5520 PRINT#1,CHR$(PEEK(12288+I));
5530 NEXTI
5540 CLOSE1
5550 RETURN
6000 PRINTCHR$(147)
6010 PRINT"DISCO O NASTRO ?"
6015 GETDV$:IFDV$<>"N"ANDDV$<>"D"THEN6015
6020 PRINT
6030 INPUT"SET NUMERO ";NS%
6070 IFDV$="N"THEN6500
6080 OPEN1,8,2,"SET #"+STR$(NS%)+",S,R"
6082 GOSUB9000
6085 IFNNTHENFORI=1TO2000:NEXT:CLOSE1:CLOSE15
```

```
6087 IFNNTHENRETURN
6090 GET#1,A$:NC%=ASC(A$):FORI=0TO(NC%+1)*8-1
6100 GET#1,A$:POKE12288+I,ASC(A$+CHR$(0))
6110 NEXTI
6120 CLOSE1:CLOSE15
6130 RETURN
6500 OPEN1,1,0,"SET #"+STR$(NS%)
6510 FORI=0TO(NC+1)%*8-1
6520 GET#1,A$:POKE12288+I,ASC(A$+CHR$(0))
6530 NEXTI
6540 CLOSE1
6550 RETURN
7000 PRINTCHR$(17)"SICURO ?"
7010 GETA$:IFA$=""THEN7010
7020 IFA$="S"THENSYS58260
7030 RETURN
8000 PRINTCHR$(147)
8010 PRINT:PRINT:PRINT:PRINT
8020 FORI=0TO15
8030 PRINTTAB(14)CHR$(C(I))CHR$(18)"         ";
8035 PRINTCHR$(154)CHR$(146)" ="I
8040 NEXT
8050 FORI=1TO3
8060 PRINTCHR$(19)
8070 PRINTTAB(13)"COLORE"I"      ";:FORJ=1TO4
8075 PRINTCHR$(20);:NEXT:INPUTA
8080 IFA<0ORA>15THEN8060
8090 IFI<>1THENPOKE53280+I,A
8100 IFI=1THENCL=A
8110 NEXT
8120 RETURN
9000 PRINT:OPEN15,8,15:INPUT#15,NN,MM$,TT,SS
9005 PRINTNN,MM$,TT,SS
9010 RETURN
10000 DATA144,5,28,159,156,30,31,158
10010 DATA129,149,150,151,152,153,154,155
```

COMMENTO A GRAF 11.

SEZIONE 1.
Aggiunte linee 30 e 40.
Linea 30: dimensiona e riempe il vettore C che conterrà i codici ASCII del colore del cursore. Ad esempio il colore di codice 2 è il rosso, il codice ASCII per ottenere il cursore rosso è 28, il vettore conterrà quindi nella seconda componente il numero 28.
Linea 40: salta alla routine di scelta dei colori (sezione 16).

SEZIONE 2.
E' stata aggiunta un' opzione: la scelta dei colori.

SEZIONE 3.
Si richiede il colore del byte.
La gestione della griglia diventa diversa.

Non si lavora più sui caratteri, ma sui colori, e non ci si sposta più di un solo passo in orizzontale, ma di due. La tecnica con cui viene gestita rimane essenzialmente la stessa.

SEZIONE 4.
Sono state tolte le opzioni REVERSATO, che con i caratteri multicolore non ha senso e RUOTATO DI 90 GRADI, data l'asimmetria dei pixel in multicolore.

SEZIONE 5.
Come prima.

SEZIONE 6.
Eliminata poichè serviva per la memorizzazione in campo inverso.

SEZIONE 7.
Il principio rimane lo stesso ma il secondo ciclo viene svolto a passi di due e vengono scambiate coppie di pixel invece che pixel singoli.

SEZIONE 8.
Come prima.

SEZIONE 9.
Eliminata poichè serviva per la memorizzazione ruotata di 90 gradi.

SEZIONE 10.
Viene richiesto il colore per il byte; come per la sezione 3 la diversità è dovuta al fatto che ora usiamo i colori invece che i caratteri nella griglia.

SEZIONE 11.
Come prima tranne che nella linea 4020 dove viene predisposto il colore richiesto per lo schermo nella routine di scelta dei colori, e nelle linee 4030-4035 dove si entra in modo multicolore.

SEZIONI 12-13-14-15.
Come prima.

SEZIONE 16.
Sono state aggiunte le linee 8000-8120.
Linea 8000: pulisce lo schermo.
Linea 8010: sposta il cursore alla quinta linea.
linea 8020: inizializza un ciclo.
Linea 8030: scrive in colonna 14 sette spazi in campo inverso del colore i-esimo usando il vettore C inizializzato nella linea 30.
Linea 8035: scrive in blu chiaro il codice colore.
Linea 8040: chiude il ciclo.
Linea 8050: inizializza un ciclo.
Linea 8060: pone il cursore in alto a sinistra.
Linea 8070: domanda il colore i-esimo, cancella la risposta precedente, accetta la risposta.
Linea 8080: controlla che il numero corrisponda a un colore, se no torna a ripetere la domanda.
Linea 8090: se il colore richiesto non è quello dello schermo lo pone nel registro appropriato.
Linea 8100: altrimenti nella variabile CL.
Linea 8110: chiude il ciclo.
Linea 8120: torna dalla routine.

2.5 PAGINA GRAFICA

Poichè il video del COMMODORE 64 è composto da 25 linee di 40 caratteri, e ciascuno di essi e formato da 8*8 punti, lo schermo è composto da ben 64000 punti, disposti in 200 righe di 320 punti. Il calcolatore ha la possibilità di gestire tutti questi punti quando viene posto nel modo PAGINA GRAFICA. Questo modo è molto utile quando vuoi disegnare grafici per applicazioni scientifiche, istogrammi per applicazioni commerciali o quando vuoi progettare giochi. Come per i caratteri, la pagina grafica può essere ad alta risoluzione o multicolore.

2.5.1 Pagina grafica ad alta risoluzione

Per entrare in modo pagina grafica è necessario porre a 1 il bit di posizione 5 del registro 17 del VIC II (indirizzo 53265 (D011H)); cioè eseguire l'istruzione: POKE 53265,PEEK(53265)OR32.
Quando il COMMODORE 64 è in modo pagina grafica, si comporta come se tutto lo schermo fosse riempito da 1000 caratteri programmabili disposti nel modo indicato nella Tabella 2.5.

```
TABELLA 2.5

  0   1   2   3 ..............  39
 40  41  42  43 ..............  79
  .   .   .   . ..............   .
  .   .   .   . ..............   .
  .   .   .   . ..............   .
  .   .   .   . ..............   .
960 961 962 963 .............. 999
```

Tabella 2.5 Posizione dei caratteri sullo schermo

L'indirizzo base di questa grande mappa dei caratteri è scritto nello stesso registro in cui è scritto l'indirizzo della normale mappa dei caratteri (vedi Paragrafo 2.3.5). Poichè la disposizione dei caratteri sullo schermo è, nel modo pagina grafica, stabilita dal calcolatore stesso, la memoria video (vedi Paragrafo 2.3.2) viene usata come una più completa memoria del colore (vedi Paragrafo 2.3.3) in cui i 4 bit più significativi danno il colore dei bit posti a 1 e i 4 meno significativi danno il colore dei bit posti a 0. Volendo quindi avere in una certa posizione dello schermo i bit a 0 del colore il cui codice sia A e i bit a 1 di colore il cui codice sia B, bisogna porre nel byte corrispondente a quella posizione il numero B*16+A.
Prova a eseguire le seguenti istruzioni:

```
PRINT CHR$(147)
POKE 53265,PEEK(53265) OR 32
POKE 53272,PEEK(53272) OR 8
```

cioè pulisci lo schermo, entra in modo pagina grafica e poni la mappa in 8192 (2000H).
Come vedi, CHR$(147) non pulisce la pagina grafica ma riempe la mappa video di 32 (ASCII di spazio), in modo pagina grafica quindi agisce sul colore e, più precisamente, pone rossi i bit a 1 e neri i bit a 0 (2*16+0=32, 2=rosso e 0=nero).
L'unico modo che abbiamo a disposizione per pulire la pagina grafica è quello di porre a 0 ogni byte della pagina stessa. Le istruzioni da eseguire sono dunque:

```
FOR I = 8192 TO 16191
POKE I,0
NEXT I
```

Purtroppo il BASIC è molto lento nell'eseguire un'operazione di questo tipo e, l'unico modo disponibile per aumentare la velocità è scrivere una piccola routine in linguaggio macchina. Riportiamo ora questa routine, GRAF12, che abbiamo fatto partire dall'indirizzo 49152 (C000H) ma che può essere trasferita senza problemi poichè contiene solamente salti con indirizzi relativi.

Nel seguito compaiono alcuni listati di programmi assembler; in essi:

- la prima colonna riporta un numero in esadecimale, preceduto da virgola, che è l'indirizzo del primo byte dell'istruzione scritta nella riga;
- le tre colonne seguenti (che possono ridursi solo a una o due) rappresentano in codice macchina, byte a byte, l'istruzione in esadecimale;
- le successive colonne rappresentano l'istruzione in linguaggio simbolico assembler (dove $ rappresenta un numero esadecimale);
- i commenti iniziano sempre con un punto e virgola.

```
GRAF12

,C000 A9 20     LDA #$20
  ;20 = prima pagina della pagina grafica
,C002 85 FF     STA $FF
  ;piu' sign. puntatore in pagina zero
,C004 A9 00     LDA #$00
,C006 85 FE     STA $FE
  ;meno sign. puntatore in pagina zero
```

```
.C008 A9 00     LDA #$00
.C00A A8        TAY
.C00B 91 FE     STA ($FE),Y
  ;Pulisce la cella Ponendovi 0
.C00D C8        INY
.C00E D0 FB     BNE $C00B
  ;se non e' finita la Pagina continua
.C010 E6 FF     INC $FF
  ;incrementa il Puntatore
.C012 A5 FF     LDA $FF
.C014 C9 40     CMP #$40
  ;40 = ult. Pag. della Pagina grafica +1
..C016 D0 F0    BNE $C008
  ;se non ha finito continua
.C018 AD 20 CB LDA $CB20
  ;52000 decimale:
  ;contiene il colore desiderato
.C01B 99 00 04 STA $0400,Y
  ;Pone nell' y-mo byte della
  ;Prima Pagina maPPa video
.C01E 99 00 05 STA $0500,Y
  ;Pone nell' y-mo byte della
  ;seconda Pagina maPPa video
.C021 99 00 06 STA $0600,Y
  ;Pone nell' y-mo byte della
  ;terza Pagina maPPa video
.C024 99 00 07 STA $0700,Y
  ;Pone nell' y-mo byte della
  ;quarta Pagina maPPa video
.C027 C8        INY
.C028 D0 F1     BNE $C01B
  ;se fine Pagina esce
.C02A 60        RTS
```

Per caricare GRAF12 nei tuoi programmi BASIC basta scrivere all'inizio del programma le istruzioni:

```
FOR I = 49152 TO 49194
READ A
POKE I,A: NEXT I
```

e scrivere le linee DATA riportate in GRAF13.

```
1 REM GRAF13
1000 DATA169,32,133,255,169,0,133,254
1010 DATA169,0,168,145,254,200,208,251
1020 DATA230,255,165,255,201,64,208,240
1030 DATA173,32,203,153,0,4,153,0
1040 DATA5,153,0,6,153,0,7,200
1050 DATA208,241,96
```

Ogni volta che vorrai usare la routine GRAF12, ti basterà inserire le linee sopra riportate, le linee DATA di GRAF13, porre in 52000 il valore desiderato del colore e dare il comando SYS 49152. Se vuoi solamente cambiare il colore dovrai dare il comando SYS 49186 (49186 (C018H)), dopo aver posto in 52000 il colore desiderato. Nota che la nostra routine pone in tutti i byte lo stesso colore anche se, per applicazioni più VARIOPINTE di quelle che illustreremo, è possibile dare a ogni matrice 8*8 un colore specifico.
Se la tua pagina grafica non parte dall'indirizzo 2000H, basta porre al posto del 32 (20H) che compare come secondo dato del programma, il numero corretto (per esempio, se la pagina grafica parte da 6000H devi porre 96 (60H) al posto di 32).
Ora che sappiamo come cancellare la pagina grafica, il problema che ci poniamo è come scriverci. Come fare cioè a DISEGNARE un solo punto tra i 64000 che formano la pagina? Cominciamo col dare un NOME ad ogni punto o, per meglio dire, due coordinate: X e Y. Chiamiamo il punto in basso a sinistra X=0,Y=0 e quello in alto a destra X=319,Y=199. Orientiamo cioè l'asse delle Y dal basso verso l'alto e l'asse delle X da sinistra verso destra. Consideriamo ora come ci appare sul video l'immagine degli 8000 byte che formano la pagina grafica: se chiamiamo OG l'indirizzo del primo byte della pagina grafica abbiamo lo schema riportato in Tabella 2.6.

Tabella 2.6

```
OG+0*320+0*8+0   OG+0*320+1*8+0   .. OG+0*320+39*8+0
OG+0*320+0*8+1   OG+0*320+1*8+1   .. OG+0*320+39*8+1
OG+0*320+0*8+2   OG+0*320+1*8+2   .. OG+0*320+39*8+2
OG+0*320+0*8+3   OG+0*320+1*8+3   .. OG+0*320+39*8+3
OG+0*320+0*8+4   OG+0*320+1*8+4   .. OG+0*320+39*8+4
```

```
OG+0*320+0*8+5  OG+0*320+1*8+5  .. OG+0*320+39*8+5
OG+0*320+0*8+6  OG+0*320+1*8+6  .. OG+0*320+39*8+6
OG+0*320+0*8+7  OG+0*320+1*8+7  .. OG+0*320+39*8+7

OG+1*320+0*8+0  OG+1*320+1*8+0  .. OG+1*320+39*8+0
OG+1*320+0*8+1  OG+1*320+1*8+1  .. OG+1*320+39*8+1
OG+1*320+0*8+2  OG+1*320+1*8+2  .. OG+1*320+39*8+2
OG+1*320+0*8+3  OG+1*320+1*8+3  .. OG+1*320+39*8+3
OG+1*320+0*8+4  OG+1*320+1*8+4  .. OG+1*320+39*8+4
OG+1*320+0*8+5  OG+1*320+1*8+5  .. OG+1*320+39*8+5
OG+1*320+0*8+6  OG+1*320+1*8+6  .. OG+1*320+39*8+6
OG+1*320+0*8+7  OG+1*320+1*8+7  .. OG+1*320+39*8+7
      .               .          .          .
      .               .          .          .

OG+21*320+0*8+0 OG+24*320+1*8+0 .. OG+24*320+39*8+0
OG+24*320+0*8+1 OG+24*320+1*8+1 .. OG+24*320+39*8+1
OG+24*320+0*8+2 OG+24*320+1*8+2 .. OG+24*320+39*8+2
OG+24*320+0*8+3 OG+24*320+1*8+3 .. OG+24*320+39*8+3
OG+24*320+0*8+4 OG+24*320+1*8+4 .. OG+24*320+39*8+4
OG+24*320+0*8+5 OG+24*320+1*8+5 .. OG+24*320+39*8+5
OG+24*320+0*8+6 OG+24*320+1*8+6 .. OG+24*320+39*8+6
OG+24*320+0*8+7 OG+24*320+1*8+7 .. OG+24*320+39*8+7
```

Tabella 2.6 Indirizzi dei byte nella pagina grafica

Come puoi vedere dalla Tabella 2.6, abbiamo per ogni byte un indirizzo del tipo:
OG+A*320+B*8+C
dove A indica la linea di caratteri contando dall'alto verso il basso (da 0 a 24) che contiene il byte desiderato, B indica la posizione del carattere nella linea (da 0 a 39) che contiene il byte e C è il numero del byte del carattere (da 0 a 7). Se vogliamo esprimere A, B e C in funzione di X e Y abbiamo:

```
A=INT((199-Y)/8)
B=INT(X/8)
C=199-Y-A*8
```

Cerchiamo di capire le relazioni precedenti. Poichè numeriamo le linee da 0 a 199, 199 è il più alto numero di linea. Sottraendo a 199 il valore della Y troviamo in quale linea di punti, contando dall'alto, si trova il punto che vogliamo disegnare; in altre parole abbiamo ribaltato l'asse delle Y. Se ora dividiamo per 8 il numero così ottenuto abbiamo, come quoziente A e come resto C. Dividendo la X per 8, si ottiene come quoziente la colonna di caratteri che contiene il byte che ci interessa cioè B; il resto di questa divisione ci dice, invece, qual'è il bit (pixel) che ci interessa. A questo punto sappiamo come annerire un punto X,Y dello schermo: basta dare al COMMODORE 64 i seguenti ordini:

```
A=INT((199-Y)/8)
B=INT(X/8)
C=199-Y-A*8
D=X-B
BY=OG+A*320+B*8+C
POKE A,PEEK(A) OR 2↑(7-D)
```

La variabile D contiene il resto della divisione di X per 8, e cioè il bit, contando da sinistra, che ci interessa. La variabile OG deve contenere già l'indirizzo del primo byte della pagina grafica. L'ultimo comando è quello che effettivamente DISEGNA il punto; pone infatti nel byte interessato il valore che ci interessa: 2 elevato a (7-D) è il numero binario formato da tutti zeri tranne il D-esimo bit a partire da sinistra. Nel byte viene posto il risultato della OR tra questo numero e il contenuto precedente del byte, per non CANCELLARE altri eventuali punti già disegnati in quel byte (vedi Paragrafo 2.2).
Purtroppo, molto spesso, in un'applicazione grafica i punti da disegnare sono moltissimi e quindi sarebbe meglio poter rendere più veloce la routine che disegna un punto sul video. Ancora una volta la soluzione è una routine in assembler, la GRAF14.

GRAF14

```
,C02B A9 C8     LDA #$C8
  ;C8 = 200 dec. = numero di punti
  ;in verticale
,C02D 38        SEC
,C02E ED 20 CB SBC $CB20
  ;sottrae la Y voluta
  ;(l'asse delle Y e' verso l'alto)
```

```
,C031 8D 20 CB STA $CB20
   ;Pone il risultato in Y
,C034 29 F8     AND #$F8
   ;azzera i 3 bit meno significativi
,C036 8D 1F CB STA $CB1F
   ;Pone il risultato in LN (linea)
,C039 AD 21 CB LDA $CB21
   ;byte meno significativo della X voluta
,C03C 29 F8     AND #$F8
   ;azzera i 3 bit meno significativi
,C03E 8D 1E CB STA $CB1E
   ;Pone il risultato in CL (colonna)
,C041 AD 20 CB LDA $CB20       ;Y
,C044 29 07     AND #$07
   ;azzera i 5 bit Piu' significativi
,C046 85 FE     STA $FE
   ;meno significativo
   ;Puntatore in Pagina zero
,C048 AD 21 CB LDA $CB21
   ;meno significativo X
,C04B 29 07     AND #$07
   ;azzera i 5 bit Piu' significativi
,C04D 8D 1D CB STA $CB1D
   ;Pone il risultato in BIT
   ;(bit che sara' "disegnato"
   ;Partendo da sinistra)
,C050 A5 FE     LDA $FE
   ;numero di byte in seno alla matrice 8*8
   ;che deve essere modificato
,C052 18        CLC
,C053 6D 1E CB ADC $CB1E
   ;aggiunge CL
,C056 85 FE     STA $FE
   ;Pone il risultato nel Puntatore
,C058 A9 20     LDA #$20
   ;Prima Pagina della Pagina grafica
,C05A 6D 22 CB ADC $CB22
   ;aggiunge Piu' significativo della X
,C05D 85 FF     STA $FF
   ;Pone il risultato nel byte
   ;Piu' signific. del Puntatore
,C05F A9 00     LDA #$00
```

```
,C061 8D 1C CB STA $CB1C
   ;azzera un byte per memoria temporanea
,C064 AD 1F CB LDA $CB1F
   ;NL
,C067 18       CLC
,C068 2A       ROL
,C069 8D 1B CB STA $CB1B
,C06C 2E 1C CB ROL $CB1C
,C06F 2E 1B CB ROL $CB1B
,C072 2E 1C CB ROL $CB1C
   ;CB1B-CB1C contiene ora NL*4
,C075 AD 1F CB LDA $CB1F
,C078 6D 1B CB ADC $CB1B
,C07B 8D 1B CB STA $CB1B
,C07E A9 00    LDA #$00
,C080 6D 1C CB ADC $CB1C
,C083 8D 1C CB STA $CB1C
   ;CB1B-CB1C contiene ora NL*5
,C086 18       CLC
,C087 2E 1B CB ROL $CB1B
,C08A 2E 1C CB ROL $CB1C
,C08D 2E 1B CB ROL $CB1B
,C090 2E 1C CB ROL $CB1C
,C093 2E 1B CB ROL $CB1B
,C096 2E 1C CB ROL $CB1C
   ;CB1B-CB1C contiene ora NL*40
,C099 18       CLC
,C09A AD 1B CB LDA $CB1B
,C09D 65 FE    ADC $FE
,C09F 85 FE    STA $FE
,C0A1 AD 1C CB LDA $CB1C
,C0A4 65 FF    ADC $FF
,C0A6 85 FF    STA $FF
   ;il puntatore contiene l'indirizzo
   ;del byte da modificare
,C0A8 A9 80    LDA #$80
   ;bit a sinistra a 1, gli altri a 0
,C0AA AE 1D CB LDX $CB1D
   ;BIT
,C0AD F0 04    BEQ $C0B3
   ;se non bisogna shiftare
   ;il contenuto di A salta
```

```
,C0AF 4A        LSR
,C0B0 CA        DEX
,C0B1 D0 FC     BNE $C0AF
  ;shifta a destra BIT volte
,C0B3 A0 00     LDY #$00
,C0B5 11 FE     ORA ($FE),Y
,C0B7 91 FE     STA ($FE),Y
  ;disegna il punto
,C0B8 60        RTS
```

Abbiamo posto l'inizio di GRAF14 in 49195 (C02BH) per poterlo tenere in memoria assieme al programma di pulizia della pagina grafica precedentemente illustrato. Anche questo programma, comunque è facilmente trasferibile poichè sono sempre stati usati salti con indirizzi relativi. Per caricare questa routine nei tuoi programmi BASIC ti basta scrivere all'inzio del programma le istruzioni:

```
FOR I = 49195 TO 49337
READ A
POKE I,A: NEXT I
```

e inserire le linee DATA riportate nel programma GRAF15.

```
1 REM GRAF15
1000 DATA169,200,56,237,32,203,141,32
1010 DATA203,41,248,141,31,203,173,33
1020 DATA203,41,248,141,30,203,173,32
1030 DATA203,41,7,133,254,173,33,203
1040 DATA41,7,141,29,203,165,254,24
1050 DATA109,30,203,133,254,169,32,109
1060 DATA34,203,133,255,169,0,141,28
1070 DATA203,173,31,203,24,42,141,27
1080 DATA203,46,28,203,46,27,203,46
1090 DATA28,203,173,31,203,109,27,203
1100 DATA141,27,203,169,0,109,28,203
1110 DATA141,28,203,24,46,27,203,46
1120 DATA28,203,46,27,203,46,28,203
1130 DATA46,27,203,46,28,203,24,173
1140 DATA27,203,101,254,133,254,173,28
```

```
1150 DATA203,101,255,133,255,169,128,174
1160 DATA29,203,240,4,74,202,208,252
1170 DATA160,0,17,254,145,254,96
```

Ogni volta che vorrai usare la routine GRAF14, ti basterà aggiungere le linee sopra riportate, le linee DATA di GRAF15, porre in 52000 il valore della Y, in 52001 il valore della X, parte meno significativa (cioè X AND 255), in 52002 il valore della X, parte più significativa (cioè X/256) e dare il comando SYS 49195.
Se la tua pagina grafica non parte dall'indirizzo 2000H, basta porre al posto del 32 (20H) che compare come quarantasettesimo dato del programma (il penultimo della sesta linea), il numero corretto (per esempio, se la pagina grafica parte da 6000H devi porre 96 (60H) al posto di 32).
Ora che abbiamo i mezzi per disegnare e cancellare abbastanza velocemente, proviamo un'applicazione con il programma GRAF16.

```
1 REM GRAF16
10 POKE53280,0:POKE53281,0
15 PRINTCHR$(147)CHR$(154)CHR$(14)CHR$(8)
20 POKE56,32:POKE55,0:CLR
30 FORI=0TO185:READA:POKE49152+I,A:NEXT
110 POKE52000,16:SYS49152
120 POKE53272,PEEK(53272)OR8
125 POKE53265,PEEK(53265)OR32
130 FORX=0TO319:I=14*π*(X/319)-7*π
135 Y=SIN(I)*I:Y=Y*90/(7*π)+100
140 GOSUB1000:NEXTX
150 GOTO150
1000 POKE52000,Y:POKE52001,XAND255
1010 POKE52002,X/256:SYS49195:RETURN
2000 DATA169,32,133,255,169,0,133,254
2010 DATA169,0,168,145,254,200,208,251
2020 DATA230,255,165,255,201,64,208,240
2030 DATA173,32,203,153,0,4,153,0
2040 DATA5,153,0,6,153,0,7,200
2050 DATA208,241,96,169,200,56,237,32
2060 DATA203,141,32,203,41,248,141,31
2070 DATA203,173,33,203,41,248,141,30
2080 DATA203,173,32,203,41,7,133,254
2090 DATA173,33,203,41,7,141,29,203
```

```
2100 DATA165,254,24,109,30,203,133,254
2110 DATA169,32,109,34,203,133,255,169
2120 DATA0,141,28,203,173,31,203,24
2130 DATA42,141,27,203,46,28,203,46
2140 DATA27,203,46,28,203,173,31,203
2150 DATA109,27,203,141,27,203,169,0
2160 DATA109,28,203,141,28,203,24,46
2170 DATA27,203,46,28,203,46,27,203
2180 DATA46,28,203,46,27,203,46,28
2190 DATA203,24,173,27,203,101,254,133
2200 DATA254,173,28,203,101,255,133,255
2210 DATA169,128,174,29,203,240,4,74
2220 DATA202,208,252,160,0,17,254,145
2230 DATA254,96
```

COMMENTO A GRAF16.
Linea 10: inizializza il video.
Linee 15-20: fine memoria a 2000H dove inizierà la pagina grafica.
Linea 30: mette in memoria, partendo dall'indirizzo 49152 (C000H), i programmi GRAF12 e GRAF14.
Linea 110: usando la routine GRAF12 pulisce la pagina grafica e pone come colore bianco su nero (bianco = 1, nero = 0 quindi 1*16+0=16 (bianco su nero)). Purtroppo, se il tuo monitor non è ottimo, data la grandezza dei punti, avrai degli strani effetti sul colore e il bianco risulterà di colori diversi in zone diverse.
Linee 120-125: entra in modo pagina grafica e pone la pagina in 8192 (2000H).
Linee 130-135: per tutte le 320 colonne, calcola il valore da dare alla Y per ottenere il grafico della funzione SIN(X)*X tra -7pigreca e 7pigreca.
Linea 140: salta alla routine 1000 e chiude il ciclo aperto nella linea 130.
Linea 150: ferma il programma. Premendo STOP/RESTORE il calcolatore torna alla normalità.
Linea 1000: routine che prepara i dati per la routine GRAF14 e la chiama. Verrà così disegnato un punto bianco in posizione X,Y.
Linee 2000-2230: contengono le linee DATA con i dati relativi alle due routine GRAF12 e GRAF14.

Come avrai notato la velocità con cui il calcolatore disegna la funzione non è delle più elevate: ciò è dovuto principalmente alla lentezza con cui il BASIC calcola il valore delle funzioni nelle linee 130-135. Se provi infatti a sostituirle con le seguenti linee:

```
130 FOR X = 0 TO 319
135 Y=X/2
```

il programma verrà svolto in metà tempo.
Il programma GRAF16 può ovviamente disegnare tutte le funzioni (che non abbiano però asintoti verticali) a patto che le linee 130-135 vengano sostituite da altre appropriate linee. Sarà comunque necessario fare ogni volta conti abbastanza noiosi per poter sfruttare tutto lo schermo e non uscirne. Poichè abbiamo un calcolatore, sarà il caso di far lavorare lui, e perciò abbiamo preparato il programma GRAF17.

```
1 REM GRAF17
5 REM SYNTAX ERROR SE FUNZIONE MAL DEFINITA
10 POKE53280,0:POKE53281,0
15 PRINTCHR$(147)CHR$(154)CHR$(14)CHR$(8)
20 POKE56,32:POKE55,0:CLR
30 FORI=0TO185:READA:POKE49152+I,A:NEXT
100 INPUT"F(X) ";F$
110 PRINTCHR$(147)CHR$(17)CHR$(17)CHR$(17);
115 PRINTCHR$(144)"140 DEFFNY(X)="F$
120 PRINT"GOTO 140"CHR$(19)
130 POKE198,2:POKE631,13:POKE632,13:END
140 DEFFNY(X)=SIN(X)
150 PRINTCHR$(154)CHR$(147)
160 INPUT"DOMINIO DA ";ID
170 INPUT"A ";FD
180 PRINT:PRINT"DISEGNO L'ASSE DELLE X ?"
190 GETA$:IFA$<>"S"ANDA$<>"N"THEN190
200 AX=1
210 IFA$="N"THENAX=0
220 PRINT:PRINT"CANCELLO LA MAPPA ?"
230 GETA$:IFA$<>"S"ANDA$<>"N"THEN230
240 CL=1
250 IFA$="N"THENCL=0
260 PRINT
265 PRINT"STO CALCOLANDO MASSIMO E MINIMO"
270 AD=FD-ID
280 FORX=0TO319
290 I=ID+X*AD/320
300 Y=FNY(I)
310 IFIR>YTHENIR=Y
320 IFFR<YTHENFR=Y
330 NEXT
```

```
340 AR=FR-IR
350 POKE52000,16:IFCLTHENSYS49152
355 SYS49176
360 POKE53272,PEEK(53272)OR8
370 POKE53265,PEEK(53265)OR32
380 ZR=ABS(IR/AR*199)
390 FORX=0TO319
400 I=ID+X*AD/320
410 Y=FNY(I)*199/AR+ZR
420 GOSUB1000
430 IFAXTHENY=ZR:GOSUB1000
440 NEXT
450 POKE198,0
460 GETA$:IFA$=""THEN460
470 POKE53272,PEEK(53272)AND247
475 POKE53265,PEEK(53265)AND223
480 PRINTCHR$(147)"VUOI UN'ALTRO GRAFICO ?"
490 GETA$:IFA$<>"S"ANDA$<>"N"THEN490
500 IFA$="N"THENSYS58260
510 RUN
1000 POKE52000,Y:POKE52001,XAND255
1010 POKE52002,X/256:SYS49195:RETURN
2000 DATA169,32,133,255,169,0,133,254
2010 DATA169,0,168,145,254,200,208,251
2020 DATA230,255,165,255,201,64,208,240
2030 DATA173,32,203,153,0,4,153,0
2040 DATA5,153,0,6,153,0,7,200
2050 DATA208,241,96,169,200,56,237,32
2060 DATA203,141,32,203,41,248,141,31
2070 DATA203,173,33,203,41,248,141,30
2080 DATA203,173,32,203,41,7,133,254
2090 DATA173,33,203,41,7,141,29,203
2100 DATA165,254,24,109,30,203,133,254
2110 DATA169,32,109,34,203,133,255,169
2120 DATA0,141,28,203,173,31,203,24
2130 DATA42,141,27,203,46,28,203,46
2140 DATA27,203,46,28,203,173,31,203
2150 DATA109,27,203,141,27,203,169,0
2160 DATA109,28,203,141,28,203,24,46
2170 DATA27,203,46,28,203,46,27,203
2180 DATA46,28,203,46,27,203,46,28
```

```
2190 DATA203,24,173,27,203,101,254,133
2200 DATA254,173,28,203,101,255,133,255
2210 DATA169,128,174,29,203,240,4,74
2220 DATA202,208,252,160,0,17,254,145
2230 DATA254,96
```

COMMENTO A GRAF17.
Linee 10-15: inizializza il video.
Linea 20: fine memoria a 2000H dove inizierà la pagina grafica.
Linea 30: mette in memoria, partendo dall'indirizzo 49152 (C000H), i programmi GRAF12 e GRAF14.
Linea 100: richiede la funzione che deve essere visualizzata. Questa viene posta nella variabile F$. Deve essere scritta secondo la normale sintassi del COMMODORE 64.
Linee 110-115: pulisce lo schermo e scrive nella quarta riga, in nero, DEFFNY(X)= seguita dalla definizione della funzione richiesta.
Linea 120: scrive sulla linea seguente il comando GOTO 140 e posiziona quindi il cursore nella posizione in alto a sinistra dello schermo.
Linea 130: scrive nel buffer della tastiera 2 RETURN (vedi il commento al programma GRAF6 del paragrafo 2.4.2) e esce. Il calcolatore scriverà quindi "READY" e si posizionerà sulla quarta linea dove: il primo RETURN inserirà nel programma la linea 140, e il secondo farà ricominciare l'esecuzione dalla linea 140.
Linea 150: pulisce nuovamente lo schermo e riporta il colore del cursore a blu chiaro.
Linee 160-170: chiede inizio e fine dell' intervallo in cui si vuole visualizzare la funzione.
Linee 180-210: se si vuole visualizzare l'asse delle ascisse la variabile AX viene posta a 1, altrimenti a 0.
Linee 220-250: se si vuole cancellare il vecchio grafico la variabile CL viene posta a 1, altrimenti a 0.
Linee 260-265: scrive il messaggio.
Linea 270: calcola l'ampiezza del dominio.
Linea 280: inizializza un ciclo.
Linee 290-300: calcola il valore della funzione su 320 punti equidistanti del dominio.
Linea 310: se la variabile IR (minimo) è maggiore del valore della funzione in questo punto, allora IR=F(I).
Linea 320: se la variabile FR (massimo) è minore del valore della funzione in questo punto, allora FR=F(I).
Linea 330: chiude il ciclo aperto nella linea 280.
Linea 340: calcola la differenza tra massimo e minimo della funzione.

Linea 350: se richiesto (CL=1) pulisce la pagina grafica e sceglie il colore bianco su nero.
Linee 360-370: entra in modo pagina grafica.
Linea 380: calcola il valore della linea di pixel che corrisponde all' asse delle ascisse.
Linee 390-440: con lo stesso procedimento con cui viene calcolato massimo e minimo, viene disegnata la funzione. Se richiesto (AX=1) nella linea 430 viene disegnato l'asse delle ascisse.
Linea 450: cancella i caratteri eventualmente contenuti nel buffer della tastiera.
Linea 460: attende che sia premuto un tasto.
Linee 470-475: ritorna al modo caratteri.
Linee 480-510: se richiesto riparte il programma, altrimenti salta alla routine di inizializzazione del BASIC.
Linea 1000: routine che prepara i dati per la routine GRAF14 e la chiama. Verrà così disegnato un punto bianco in posizione X,Y.
Linee 2000-2230: contengono i dati relativi alle due routine GRAF12 e GRAF14.

Se si presta un pò di attenzione è possibile disegnare anche funzioni con asintoti verticali: basta infatti fare in modo che il COMMODORE 64 non debba calcolare il valore della funzione nel punto in cui questa non è definita. Ad esempio volendo avere il grafico della funzione Y=1/X in un intorno dello 0 dovrai farla disegnare su un dominio che va, ad esempio, da -1 a 1.1, in modo che il valore più vicino allo 0, per cui il calcolatore calcolerà la funzione, sarà -2.49999994 E-3 per X = 153.
I risultati che si possono ottenere con GRAF17 sono già abbastanza simpatici ma il prossimo programma, GRAF18, intende dare risultati molto più spettacolari: disegna funzioni reali di due variabili reali, disegna cioè figure tridimensionali.

```
1 REM GRAF18
10 POKE56,32:POKE55,0:CLR
20 DIMX%(319),N%(319)
25 FORI=0TO319:N%(I)=200:NEXT
30 FORI=0TO185:READA:POKE49152+I,A:NEXT
40 POKE53280,0:POKE53281,0
50 POKE53272,PEEK(53272)OR8
60 POKE53265,PEEK(53265)OR32
70 POKE52000,16:SYS49152
80 X3=-2:FORY3=-2TO2STEP.02:GOSUB240:NEXT
90 SP=.3
100 FORX3=-2TO2STEP.02:Y3=-2
105 GOSUB240:Y3=2:GOSUB240
110 RX=X3-INT(X3/SP)*SP
```

```
120 FORY3=-2+RXTO2STEPSP
130 GOSUB240
140 NEXTY3
150 FORY3=2-RXTO-2STEP-SP
160 GOSUB240
170 NEXTY3:NEXTX3
180 X3=2:FORY3=-2TO2STEP.02:GOSUB240:NEXT
190 POKE198,0
200 GETA$:IFA$=""THEN200
210 POKE53272,PEEK(53272)AND247
220 POKE53265,PEEK(53265)AND223
230 PRINTCHR$(147):END
240 Z3=Y3*Y3/4-X3*X3/4
245 X2=160+(Y3+X3/2)*49:Y2=100+(Z3+X3/2)*49
250 IFY2>X%(X2)THENX%(X2)=Y2:GOTO280
260 IFY2<N%(X2)THENN%(X2)=Y2:GOTO290
270 RETURN
280 IFY2<N%(X2)THENN%(X2)=Y2
290 GOSUB300:RETURN
300 POKE52000,Y2:POKE52001,X2AND255
310 POKE52002,X2/256:SYS49195:RETURN
1000 DATA169,32,133,255,169,0,133,254
1010 DATA169,0,168,145,254,200,208,251
1020 DATA230,255,165,255,201,64,208,240
1030 DATA173,32,203,153,0,4,153,0
1040 DATA5,153,0,6,153,0,7,200
1050 DATA208,241,96,169,200,56,237,32
1060 DATA203,141,32,203,41,248,141,31
1070 DATA203,173,33,203,41,248,141,30
1080 DATA203,173,32,203,41,7,133,254
1090 DATA173,33,203,41,7,141,29,203
1100 DATA165,254,24,109,30,203,133,254
1110 DATA169,32,109,34,203,133,255,169
1120 DATA0,141,28,203,173,31,203,24
1130 DATA42,141,27,203,46,28,203,46
1140 DATA27,203,46,28,203,173,31,203
1150 DATA109,27,203,141,27,203,169,0
1160 DATA109,28,203,141,28,203,24,46
1170 DATA27,203,46,28,203,46,27,203
1180 DATA46,28,203,46,27,203,46,28
1190 DATA203,24,173,27,203,101,254,133
```

```
1200 DATA254,173,28,203,101,255,133,255
1210 DATA169,128,174,29,203,240,4,74
1220 DATA202,208,252,160,0,17,254,145
1230 DATA254,96
```

COMMENTO A GRAF18.
Linea 10: pone la fine della memoria a 8192 (2000H).
Linea 20: dimensiona i vettori X% (max) e N% (min) di 320 elementi, quante sono le colonne di punti nel video. Questi serviranno a sapere se il punto che si stà per disegnare è coperto dalla funzione o no. Dopo aver dimensionato i vettori pone in tutti gli elementi di N% il numero 200.
Linea 30: carica in memoria le routine GRAF12 e GRAF14.
Linea 40: schermo e sfondo neri.
Linea 50: entra nel modo pagina grafica.
Linea 60: pone la pagina grafica in 8192 (2000H).
Linea 70: sceglie bianco su nero e salta alla routine GRAF12.
Linea 80: calcola e disegna in assonometria (appoggiandosi alla routine in 240) il valore della funzione sul segmento X=-2, -2 < Y < 2, cioè il segmento più vicino all'osservatore. I punti del segmento su cui viene fatto il conto sono molto vicini (passo=0.02 cioè 1/50 del segmento stesso).
Linea 90: pone il passo del reticolo=0.3. Aumentando il valore della variabile SP si otterrà un reticolo più largo, diminuendolo un reticolo più stretto.
Linee 100-105: inizializza un ciclo che incrementa la X (asse che va dall'osservatore verso lo schermo) con passo di 0.02. Quindi calcola e disegna il valore della funzione per Y=-2 e per Y=2 cioè ai due lati del dominio.
Linee 110-120: inizializza un ciclo che incrementa la Y con passo SP (in questo caso 0.3) partendo da un valore di Y compreso tra -2-SP e -2+SP in modo che il reticolo sia diagonale rispetto al dominio.
Linea 130: calcola e disegna il valore della funzione per quei valori di X e Y.
Linea 140: chiude il ciclo della Y.
Linee 150-170: esegue lo stesso lavoro per i valori della Y simmetrici per disegnare le diagonali perpendicolari alle prime.
Linea 170: chiude anche il ciclo della X.
Linea 180: come la linea 80 solo che il segmento è X=2 (il più lontano dall'osservatore).
Linea 190: svuota il buffer di tastiera.
Linea 200: attende che sia premuto un tasto.
Linee 210-220: riportano il video in modo caratteri.
Linea 230: pulisce lo schermo e termina il programma.
Linee 240-245: calcola il valore della funzione tridimensionale (Z3) e quindi i valori delle X e Y bidimensionali cioè quelli dell'assonometria.

Linea 250: se il valore massimo della Y2 su quella colonna è minore del valore appena calcolato, allora pone il valore del massimo sulla colonna uguale al valore appena calcolato, e salta alla linea 280.
Linea 260: se il valore minimo della Y2 su quella colonna è maggiore del valore appena calcolato, allora pone il valore del minimo sulla colonna uguale al valore appena calcolato, e salta alla linea 290.
Linea 280: se il valore della Y2 cade tra massimo e minimo il punto è nascosto dalla funzione precedentemente disegnata e quindi non deve essere disegnato.
Linea 290: il programma arriva qui se il valore della Y è maggiore del massimo della Y su quella colonna. Qui, se il valore della Y è anche minore del minimo su questa colonna pone minimo uguale al valore appena calcolato. Questo caso si verifica quando si disegna il primo punto su una colonna: infatti all'inizializzazione: N%=200 e X%=0. Salta alla routine che disegna il punto e ritorna.
Linee 300-310: prepara la routine GRAF14 e la chiama, quindi ritorna.
Linee 1000-1230: contengono i dati relativi alle routine GRAF12 e GRAF14.

Come hai visto questo programma permette di disegnare una porzione di funzione tridimensionale e, per essere più precisi disegna una porzione di spazio $-2<X<2$, $-2<Y<2$, $-1<Z<1$. Se vuoi vedere una funzione in uno spazio maggiore o minore di questo conviene modificare la funzione stessa nelle linee 240-245. Ad esempio, se si vuole vedere la funzione F(X,Y) su un dominio $-10<X<10$, $-20<Y<20$, basterà impostare la funzione F(X*5,Y*10)/M, dove M è il massimo che ABS(F(X*5,Y*10)) assume nell'intervallo $-2<X<2$, $-2<Y<2$, o (che è la stessa cosa) che la funzione F(X,Y) assume nell'intervallo $-10<X<10$, $-20<Y<20$. In ogni caso la funzione sarà centrata nell'origine (X=0,Y=0). Se preferisci puoi modificare il programma, tenendo conto delle regole appena mostrate, approssimativamente come e stato fatto per il programma GRAF16: l'unico svantaggio che può comportare un'operazione di questo tipo è il tempo che impiegherà il programma per calcolare massimo e minimo.
Per ottenere figure diverse da quella che si ottiene normalmente, prova a far girare il programma sostituendo la linea 240 con una delle seguenti:

```
240 Z3=SIN(Y3*2)
```

oppure:

```
240 Z3=SIN(X3*X3+Y3*Y3)
```

oppure.

```
240 Z3=(X3*X3+Y3*Y3)*(X3*X3+Y3*Y3)/32-1
```

Riportiamo nelle Figure 2.1, 2.2, 2.3 e 2.4 i grafici ottenuti con il programma GRAF18, come appare nel listato, e con le 3 modifiche della linea 240 sopra indicate.

240 Z3=Y3*Y3/4-X3*X3/4

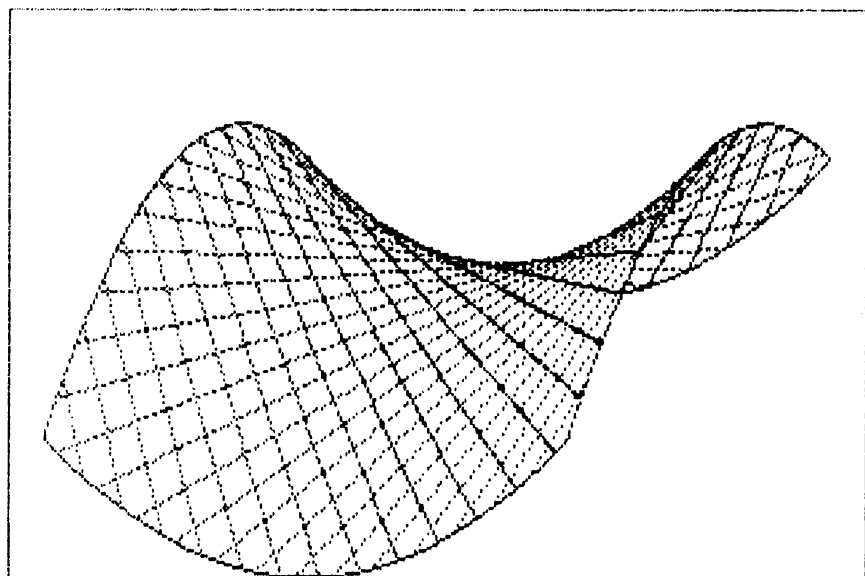



Figura 2.1 Grafico ottenuto con il programma GRAF18

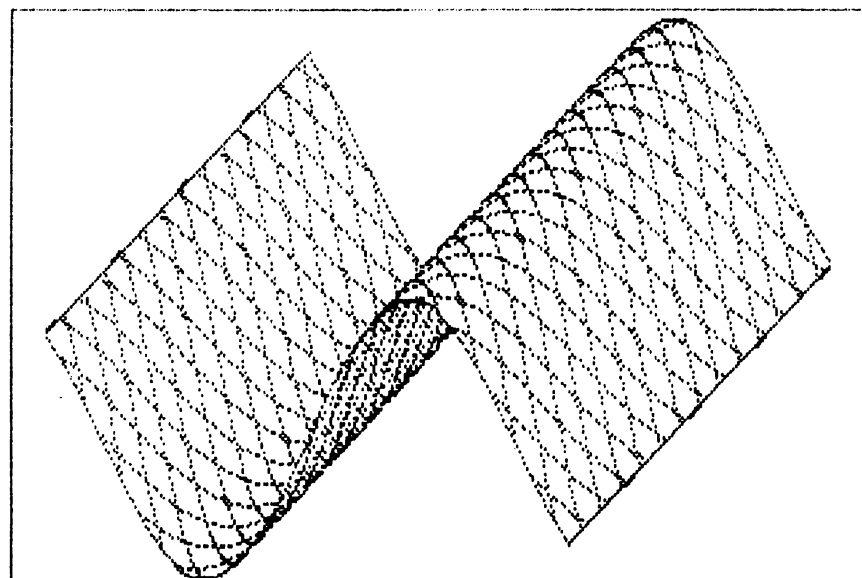



Figura 2.2 Grafico ottenuto con il programma GRAF18 con modifica 1

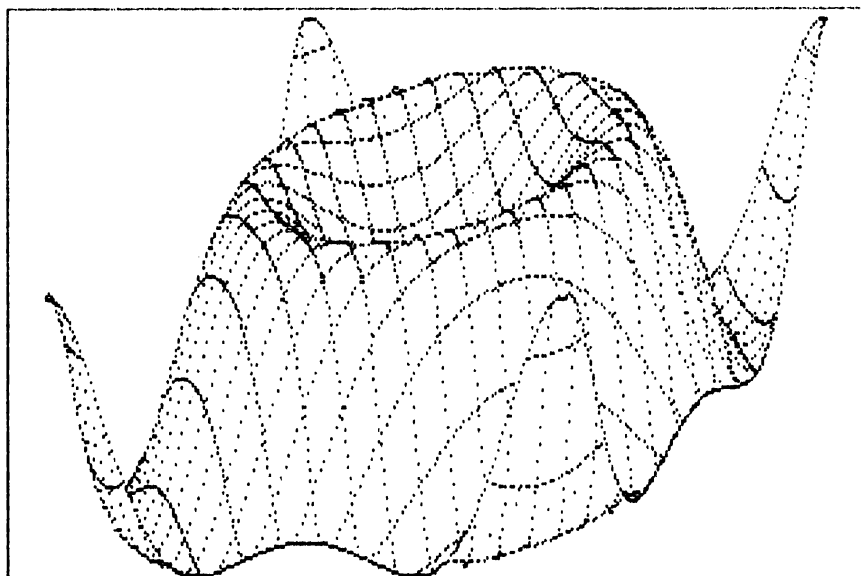



Figura 2.3 Grafico ottenuto con il programma GRAF18 con modifica 2

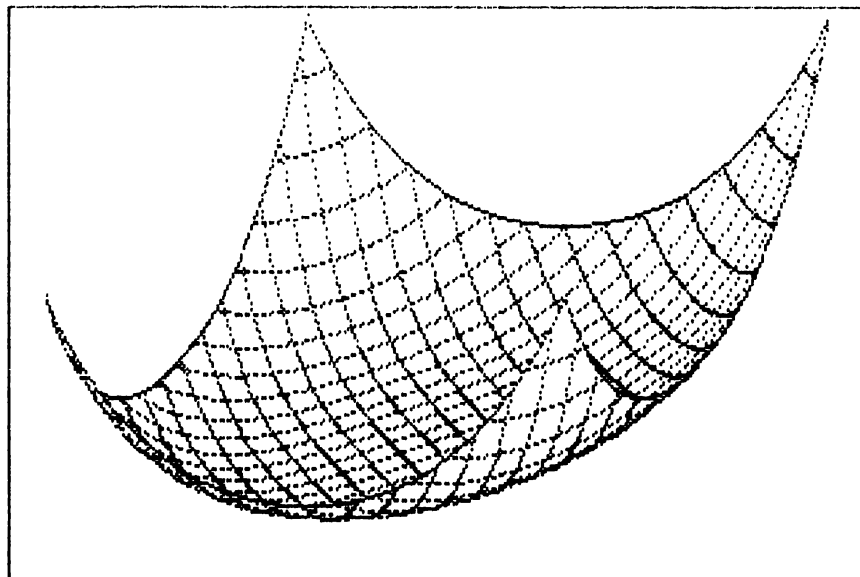



Figura 2.4 Grafico ottenuto con il programma GRAF18 con modifica 3

2.5.2 Pagina grafica multicolore

Come per i caratteri, anche per la pagina grafica nel modo multicolore ogni punto dello schermo corrisponde a 2 bit della memoria e la risoluzione orizzontale viene quindi dimezzata (i punti sono ora SOLO 32000 disposti su 200 linee di 160 punti). Come per i caratteri, anche per la pagina grafica, il comando da dare al COMMODORE 64 perchè entri in modo multicolore è:

POKE 53270,PEEK(53270) OR 16

bisogna cioè porre a 1 il bit di posizione 4 del registro 22 (16H) del VIC II. La differenza che esiste tra il modo multicolore con i caratteri e il modo multicolore con la pagina grafica è nella scelta dei colori: nel modo pagina grafica ogni matrice 4*8 può avere, indipendentemente dalle altre, 3 colori diversi scelti tra i 16 disponibili e un solo colore in comune con le altre, anch'esso scelto tra i 16 a disposizione. Il colore comune a tutte le matrici è quello dello schermo, mentre gli altri 3 vengono posti:

A) nei 4 bit più significativi della memoria video (che, come abbiamo già visto, viene utilizzata come mappa del colore, quando il COMMODORE 64 è in modo pagina grafica).

B) nei 4 bit meno significativi della mamoria video.

C) nella mappa del colore.

Per la matrice 4*8 posta in alto a sinistra vale quindi quanto riportato nella Tabella 2.7.

```
Tabella 2.7

BIT     COLORE DEL PUNTO CORRISPONDENTE

00      colore dello schermo
01      colore contenuto nei 4 bit piu'
        significativi  del  primo  byte
        della memoria video
10      colore contenuto nei 4 bit meno
        significativi  del  primo  byte
        della memoria video
11      colore contenuto nel primo byte
        della memoria del colore
```

Tabella 2.7 Corrispondenze tra COPPIE di BIT e COLORI in multicolore

Usando le routine che abbiamo già visto per la gestione della pagina grafica, si può facilmente gestire anche la pagina grafica multicolore: eccone un esempio nel programma GRAF19.

```
 1 REM GRAF19
 10 POKE56,32:POKE55,0:CLR
 20 DIMX%(159),N%(159)
 25 FORI=0TO159:N%(I)=200:NEXT
 30 FORI=0TO185:READA:POKE49152+I,A:NEXT
 40 POKE53280,0:POKE53281,0
 50 POKE53272,PEEK(53272)OR8
 55 POKE53270,PEEK(53270)OR16
 60 POKE53265,PEEK(53265)OR32
 70 POKE52000,39:SYS49152
 80 X3=-2:FORY3=-2TO2STEP.02:GOSUB240:NEXT
 90 SP=.3
 100 FORX3=-2TO2STEP.02:Y3=-2
 105 GOSUB240:Y3=2:GOSUB240
 110 RX=X3-INT(X3/SP)*SP
 120 FORY3=-2+RXTO2STEPSP
 130 GOSUB240
 140 NEXTY3
 150 FORY3=2-RXTO-2STEP-SP
 160 GOSUB240
 170 NEXTY3:NEXTX3
 180 X3=2:FORY3=-2TO2STEP.02:GOSUB240:NEXT
 190 POKE198,0
 200 GETA$:IFA$=""THEN200
 210 POKE53272,PEEK(53272)AND247
 215 POKE53270,PEEK(53270)AND239
 220 POKE53265,PEEK(53265)AND223
 230 PRINTCHR$(147):END
 240 Z3=SIN(X3*3)
 245 X2=INT(80+(Y3+X3/2)*24):Y2=100+(Z3+X3/2)*49
 250 IFY2>X%(X2)THENX%(X2)=Y2:CL=0:GOTO280
 260 IFY2<N%(X2)THENN%(X2)=Y2:CL=1:GOTO300
 270 RETURN
 280 IFY2<N%(X2)THENN%(X2)=Y2
 290 GOSUB300:RETURN
 300 IFCLTHENX=X2*2+1:GOSUB305:RETURN
```

```
302 X=X2*2
305 POKE52000,Y2:POKE52001,XAND255
310 POKE52002,X/256:SYS49195:RETURN
1000 DATA169,32,133,255,169,0,133,254
1010 DATA169,0,168,145,254,200,208,251
1020 DATA230,255,165,255,201,64,208,240
1030 DATA173,32,203,153,0,4,153,0
1040 DATA5,153,0,6,153,0,7,200
1050 DATA208,241,96,169,200,56,237,32
1060 DATA203,141,32,203,41,248,141,31
1070 DATA203,173,33,203,41,248,141,30
1080 DATA203,173,32,203,41,7,133,254
1090 DATA173,33,203,41,7,141,29,203
1100 DATA165,254,24,109,30,203,133,254
1110 DATA169,32,109,34,203,133,255,169
1120 DATA0,141,28,203,173,31,203,24
1130 DATA42,141,27,203,46,28,203,46
1140 DATA27,203,46,28,203,173,31,203
1150 DATA109,27,203,141,27,203,169,0
1160 DATA109,28,203,141,28,203,24,46
1170 DATA27,203,46,28,203,46,27,203
1180 DATA46,28,203,46,27,203,46,28
1190 DATA203,24,173,27,203,101,254,133
1200 DATA254,173,28,203,101,255,133,255
1210 DATA169,128,174,29,203,240,4,74
1220 DATA202,208,252,160,0,17,254,145
1230 DATA254,96
```

Il programma GRAF19 differisce dal precedente solo per pochissime cose che gli permettono di disegnare la funzione 3-D con la parte superiore di un colore e quella inferiore di un altro. Vediamo queste differenze nel commento.

COMMENTO A GRAF19.
Linea 20: i vettori N% e X% hanno solo 160 elementi poichè la risoluzione orizzontale è dimezzata.
Linea 55: entra anche in modo multicolore.
Linea 70: i colori scelti sono rosso e giallo.
Linee 240-245: abbiamo scelto una funzione diversa e il calcolo della X2 è diverso a causa della diversa risoluzione del video.
Linee 250-260: se il punto viene disegnato sopra, la variabile CL viene posta a 0 altrimenti a 1.
Linea 305: è la vecchia 300.

Linea 300: se CL=1 annerisce il bit meno significativo della coppia di bit che ci interessa.
Linea 302: se CL=0 annerisce il bit più significativo della coppia di bit che ci interessa.

Anche in questo caso puoi sostituire la linea 240 con una di quelle suggerite per modificare il programma GRAF18 e provare il programma.

2.6 ALTRE POSSIBILITA' DEL VIC II

Vediamo ora le caratteristiche del VIC II che non abbiamo ancora considerato.

2.6.1 Annullamento dello schermo (Screen Blanking)

E' possibile annullare tutto ciò che viene visualizzato sullo schermo ponendo a zero il bit di posizione 4 del registro 17 (11H) del VIC II. Tale registro risponde all'indirizzo 53265 (D011H): il comando da dare al COMMODORE 64 è quindi:
POKE 53265,PEEK(53265) AND 239.
Compiendo questa operazione lo schermo diventa completamente vuoto e dello stesso colore del bordo (come quando il calcolatore carica da nastro). Per riportare la situazione alla normalità bisogna compiere l'operazione inversa: dare cioè il comando:
POKE 53265,PEEK(53265) OR 16.
Si può sfruttare questa opzione del VIC II per compiere delle operazioni sullo schermo mentre questo non si può vedere, ma il vero significato di questo bit è un altro.
Come sai, sia il VIC II che la CPU accedono alla memoria. Come fanno a non litigare mai? Cioè, come è possibile che non vogliano leggere entrambi, nello stesso momento, due byte di memoria diversi? Ciò sarebbe impossibile perchè la memoria accetta un solo indirizzo alla volta e può presentare o accettare un solo dato alla volta. Esiste un segnale importantissimo all'interno del calcolatore che temporizza tutto il sistema e prende il nome di CLOCK. Questo segnale non è altro che un'onda quadra di frequenza ben precisa (nel nostro caso 0.98 MHz); cioè il clock, ogni secondo, passa dallo stato ALTO allo stato BASSO (o viceversa) 980000 volte. La CPU, che ha un piedino che è collegato al clock, è stata costruita in modo che acceda alla memoria solamente quando il clock è alto. Al VIC II rimangono quindi, per leggere la memoria tutti i periodi in cui il clock è basso. Poichè il clock rimane basso per 1/1960000 di secondo (cioè per poco più di mezzo microsecondo), ogni volta che il clock è basso il VIC II ha tempo per fare una sola lettura: potrà quindi leggere un dato ogni microsecondo. Ma in alcuni casi il VIC II, per eseguire

correttamente le sue funzioni, deve leggere dati dalla memoria con una frequenza più elevata e deve quindi chiedere alla CPU di leggere la memoria anche quando il clock è alto, rubandole tempo. Il nostro famoso bit quindi regala tempo alla CPU visto che il VIC II, non dovendo più disegnare nè il contenuto del quadro nè gli sprite, non chiederà più tempo alla CPU quando non gli spetta. Prova a far girare il programma GRAF20.

```
1 REM GRAF20
10 POKE53265,PEEK(53265)AND239
20 TI$="000000"
30 FORI=1TO10000:NEXT
40 A=TI/60
50 POKE53265,PEEK(53265)OR16
60 TI$="000000"
70 FORI=1TO10000:NEXT
80 B=TI/60
90 PRINT"CON SCHERMO ANNULLATO :"A"SECONDI"
100 PRINT"CON SCHERMO NON ANNULLATO :"B"SECONDI"
```

Come vedi il tempo impiegato a schermo annullato è minore di quello impiegato a schermo pieno. Un altro, più importante vantaggio di questa opzione del VIC II consiste nelle routine di ritardo: in tali routine, se scritte in linguaggio macchina, il ritardo può essere calcolato al milionesimo di secondo annullando lo schermo e disabilitando gli interrupt. Questi ritardi cosi precisi servono nel caso di alcune operazioni di I/O come, ad esempio, quelle del nastro.

2.6.2 Il registro di linea e i registri di interrupt

Il registro di linea è un registro di 9 bit, i cui 8 meno significativi sono posti nel registro 18 del VIC II (indirizzo 53266 (D012H)) e il più significativo è il bit di posizione 7 del registro 17 (indirizzo 53265 (D011H)). Lo scopo di questo registro è duplice: quando viene letto dà la posizione della linea del pennello elettronico del televisore. Quando vi si scrive (compreso il bit più significativo) il numero viene memorizzato e quando la linea diventa uguale al numero scritto il VIC II lancia un INTERRUPT (IRQ).

Per sapere come gestire questo interrupt bisogna conoscere il significato di due registri molto importanti del VIC II: il registro di stato degli interrupt e il registro di abilitazione degli interrupt.

Il REGISTRO DI STATO DEGLI INTERRUPT rispode all'indirizzo 53273 (D019H) e svolge due funzioni come il registro di linea; quando viene letto segnala quale interrupt (di quelli che possono essere generati dal VIC II) è pendente.

La segnalazione avviene in questo modo:

- bit di posizione 0: se è a 1, è pendente l'interrupt generato dalla funzione di linea;
- bit di posizione 1: se è a 1, è pendente l'interrupt generato da una collisione tra sprite e dati;
- bit di posizione 2: se è a 1, è pendente l'interrupt generato da una collisione tra sprite;
- bit di posizione 3: se è a 1, è pendente l'interrupt generato da una transizione negativa della light pen;
- bit di posizione 7: è a 1 ogni volta che uno dei 4 interrupt è pendente.

Quando, invece, si scrive nel registro di stato degli interrupt, si avvisa il VIC II che si è pronti a ricevere un altro interrupt dello stesso tipo.
Ad esempio, se due sprite si scontrano, viene lanciato un interrupt alla CPU e viene posto a 1 il bit di posizione 2 del registro di stato degli interrupt. Quando la CPU è pronta ad accettare un altro interrupt scrive 1 nello stesso bit dello stesso registro, avvisando così il VIC II che può generare un altro interrupt.

Il REGISTRO DI ABILITAZIONE DEGLI INTERRUPT risponde all'indirizzo 53274 (D01AH) e ha lo stesso formato del registro di stato degli interrupt. Se un bit di questo registro è a 1, l'interrupt corrispondente è abilitato, se è a 0, quando si verifica una condizione tra quelle sopra esposte, non viene chiamato alcun interrupt, ma viene ugualmente posto a 1 il bit corrispondente del registro di stato degli interrupt.
Ovviamente, per poter usufruire di questa versatile struttura di interrupt bisogna programmare in linguaggio macchina.
Il programma GRAF21, che segue, è un esempio istruttivo di come possano essere usati i registri di linea e di interrupt.

GRAF21

```
,C000 78          SEI
  ;disabilita l'interrupt
,C001 A9 C0       LDA #$C0
,C003 8D 15 03    STA $0315
,C006 A9 26       LDA #$26
```

```
,C008 8D 14 03 STA $0314
  ;la nuova routine di interrupt
  ;parte ora da C026
,C00B A9 FA    LDA #$FA
,C00D 8D 12 D0 STA $D012
,C010 AD 11 D0 LDA $D011
,C013 29 7F    AND #$7F
,C015 8D 11 D0 STA $D011
  ;il VIC II lancera' un interrupt quando
  ;il raster conterra' $FA (fine schermo)
,C018 AD 1A D0 LDA $D01A
,C01B 09 01    ORA #$01
,C01D 8D 1A D0 STA $D01A
  ;abilita l'interrupt del raster
,C020 A9 FF    LDA #$FF
,C022 85 FF    STA $FF
  ;inizializza il flag che avvisa in che
  ;posizione dello schermo e' stato
  ;lanciato l'interrupt
,C024 58       CLI
  ;accetta gli interrupt
,C025 60       RTS
  ;torna al BASIC
,C026 AD 19 D0 LDA $D019
  ;NUOVA ROUTINE DI INTERRUPT
,C029 29 01    AND #$01
,C02B D0 03    BNE $C030
  ;se l'interrupt e' stato chiamato
  ;da VIC II salta
,C02D 4C 31 EA JMP $EA31
  ;altrimenti va alla normale routine
  ;di interrupt
,C030 A9 01    LDA #$01
,C032 8D 19 D0 STA $D019
  ;avvisa il VIC II che ha accettato
  ;l'interrupt
,C035 AD 20 D0 LDA $D020
,C038 49 01    EOR #$01
,C03A 8D 20 D0 STA $D020
  ;cambia colore al bordo
,C03D A5 FF    LDA $FF
,C03F D0 10    BNE $C051
```

```
  ;se non siamo all'inizio dello
  ;schermo salta
,C041 A9 FA    LDA #$FA
,C043 8D 12 D0 STA $D012
,C046 AD 11 D0 LDA $D011
,C049 29 7F    AND #$7F
,C04B 8D 11 D0 STA $D011
  ;chiede il prossimo interrupt
  ;alla linea $FA (fine schermo)
,C04E 18       CLC
,C04F 90 0D    BCC $C05E
  ;salta sempre al ritorno da interrupt
  ;(non abbiamo usato un'istruzione JMP
  ;per rendere il programma rilocabile)
,C051 A9 32    LDA #$32
,C053 8D 12 D0 STA $D012
,C056 AD 11 D0 LDA $D011
,C059 29 7F    AND #$7F
,C05B 8D 11 D0 STA $D011
  ;chiede il prossimo interrupt alla
  ;linea $32 (inizio schermo)
,C05E A9 FF    LDA #$FF
,C060 45 FF    EOR $FF
,C062 85 FF    STA $FF
  ;cambia il flag
,C064 68       PLA
,C065 A8       TAY
,C066 68       PLA
,C067 AA       TAX
,C068 68       PLA
,C069 40       RTI
  ;torna da interrupt
```

Per chi non avesse la possibilita di porre facilmente in memoria il programma in assembler presentiamo la versione realizzata con linee DATA, GRAF22.

```
1 REM GRAF22
10 FORI=49152TO49257:READA:POKEI,A:NEXT
20 SYS49152:NEW
1000 DATA120,169,192,141,21,3,169,38
1010 DATA141,20,3,169,250,141,18,208
1020 DATA173,17,208,41,127,141,17,208
1030 DATA173,26,208,9,1,141,26,208
1040 DATA169,255,133,255,88,96,173,25
1050 DATA208,41,1,208,3,76,49,234
1060 DATA169,1,141,25,208,173,32,208
1070 DATA73,1,141,32,208,165,255,208
1080 DATA16,169,250,141,18,208,173,17
1090 DATA208,41,127,141,17,208,24,144
1100 DATA13,169,50,141,18,208,173,17
1110 DATA208,41,127,141,17,208,169,255
1120 DATA69,255,133,255,104,168,104,170
1130 DATA104,64
```

Il risultato del programma GRAF22 è un quadro video un pò fuori dal comune, con la parte orizzontale del bordo di un colore e quella verticale di un altro colore. Questo effetto è stato ottenuto facendo in modo che il VIC II lanci un segnale di interrupt ogni volta che, nel disegnare il quadro video, il pennello elettronico è arrivato dove inizia la parte centrale dello schermo; a questo punto viene cambiato il colore del bordo. Riportiamo il colore del bordo al suo valore primitivo, con lo stesso procedimento, quando il pennello elettronico è arrivato alla fine della parte centrale dello schermo.

2.6.3 Scorrimento fine (Smooth Scrolling)

Il VIC II permette di far scorrere tutto lo schermo di un solo punto (un ottavo di carattere) sia in direzione verticale che orizzontale.
I registri con cui si controllano queste operazioni sono rispettivamente i registri 17 e 22 del VIC II che rispondono agli indirizzi 53265 (D011H) e 53270 (D016H). I tre bit meno significativi di questi registri indicano quale delle 8 possibili posizioni (da 0 a 7) debbano assumere i caratteri sul video, mentre il bit 3 di questi registri

seleziona, se posto a 0, rispettivamente il modo a 24 righe (normalmente sono 25) e a 38 colonne (normalmente sono 40).
Nota che lo schermo rimane sempre di 25 righe di 40 colonne, ma ponendo a zero quei bit vengono visualizzate solamente 24 righe e 38 colonne: questo permette di aggiungere, nella parte nascosta dello schermo, i dati nuovi e farli entrare lentamente nella parte visibile usando lo scorrimento fine. Per uno scorrimento fine devi:

1) diminuire la parte visibile dello schermo nella direzione in cui lo vuoi far scorrere (ad esempio, se vuoi far scorrere lo schermo in direzione verticale, lo dovrai portare a 24 righe);

2) porre i 3 bit meno significativi del registro interessato al valore massimo o minimo (dipende se lo scorrimento viene fatto dall' alto verso il basso o viceversa, da sinistra verso destra o da destra verso sinistra);

3) scrivere i dati nella parte nascosta dello schermo;

4) incrementare (o decrementare) il registro fino al valore massimo (o minimo);

5) riportare il registro al valore del passo due e far scorrere tutto lo schermo di un carattere intero nella direzione dello scorrimento; questa operazione deve essere fatta con una routine in linguaggio macchina, poichè in BASIC si vede nettamente che questa operazione equivale a fare 7 passi indietro più 8 in avanti,mentre deve sembrare sempre che sia un solo passo in avanti;

6) tornare al punto 3.

Vediamo un paio di esempi, che chiariscano meglio questi concetti, con i programmi GRAF23 e GRAF24.

```
1 REM GRAF23
10 POKE53280,0:POKE53281,0
15 PRINTCHR$(147)CHR$(154)
20 PRINT"CHE SCRITTA VUOI FAR SCORRERE ":PRINT
30 GETA$:IFA$=CHR$(13)ORLEN(B$)=253THEN50
40 B$=B$+A$:PRINTA$;:GOTO30
50 B$=B$+"   "
60 L=LEN(B$):PRINTCHR$(147)
65 POKE53270,PEEK(53270)AND247
70 FORI=1TOLEN(B$)
80 PRINTCHR$(19)" ";
100 PRINTCHR$(20);
103 POKE53270,(PEEK(53270)AND248)+7
```

```
105 PRINTCHR$(17)CHR$(157)MID$(B$,I,1)
110 FORJ=6TO0STEP-1
120 POKE53270,(PEEK(53270)AND248)+J
130 NEXT
140 NEXT
150 GOTO70
```

COMMENTO A GRAF23.
Linee 10-15: inizializza il video.
Linea 20: scrive la domanda.
Linee 30-40: ricevono una stringa che può essere lunga fino a 253 caratteri.
Linea 50: aggiunge 2 spazi alla stringa.
Linee 60-65: calcola la lunghezza della stringa, pulisce lo schermo e seleziona il modo a 38 colonne.
Linea 70: inizializza un ciclo.
Linea 80: posiziona il cursore sulla seconda colonna della prima linea.
Linea 100: cancella un carattere in modo da far scorrere tutta la prima linea a sinistra e riposizionare il cursore nell'angolo in alto a sinistra, pone 7 nei tre bit meno significativi del registro dello scorrimento fine orizzontale (7 = MAX a destra),sposta il cursore in basso e quindi a sinistra per posizionarlo sull'ultimo carattere della prima linea dove stampa l'i-esimo carattere della stringa.
Linee 110-130: ciclo che decrementa il contenuto dei 3 bit meno significativi del registro di scorrimento orizzontale fino a 0, spostando così le scritte gradualmente fino al massimo a sinistra.
Linea 140: chiude il ciclo iniziato nella linea 70.
Linea 150: salta nuovamente all'inizio del ciclo e ripete fino a che il programma non venga fermato con STOP o STOP/RESTORE.

Come vedi un programma completamente BASIC, non dà dei buoni risultati. La linea 100 andrà quindi sostituita con un'adeguata routine in linguaggio macchina, come la GRAF24, di cui riportiamo il listato commentato.

GRAF24

```
,C000 78          SEI
  ;disabilita l'interrupt
,C001 AD 12 D0 LDA $D012
,C004 C9 50    CMP #$50
,C006 D0 F9    BNE $C001
```

```
,C008 AD 11 D0 LDA $D011
,C00B 29 80    AND #$80
,C00D D0 F2    BNE $C001
  ;attende che il registro raster
  ;contenga $50
,C00F A9 14    LDA #$14
  ;carica nell'accumulatore $14 = 20
  ;ASCII di delete
,C011 20 CA F1 JSR $F1CA
  ;salta alla routine di output a video
,C014 AD 16 D0 LDA $D016
,C017 09 07    ORA #$07
,C019 8D 16 D0 STA $D016
  ;pone 7 nel registro dello smooth
  ;scrolling (7 = il massimo a destra)
,C01C 58       CLI
  ;riabilita l'interrupt
,C01D 60       RTS
  ;torna al basic
```

Il programma GRAF23, per lo scorrimento orizzontale, diventerà quindi il GRAF25.

```
1 REM GRAF25
5 FORI=49152TO49181:READA:POKEI,A:NEXT
10 POKE53280,0:POKE53281,0
15 PRINTCHR$(147)CHR$(154)
20 PRINT"CHE SCRITTA VUOI FAR SCORRERE ":PRINT
30 GETA$:IFA$=CHR$(13)ORLEN(B$)=253THEN50
40 B$=B$+A$:PRINTA$;:GOTO30
50 B$=B$+"   "
60 L=LEN(B$):PRINTCHR$(147)
65 POKE53270,PEEK(53270)AND247
70 FORI=1TOLEN(B$)
80 PRINTCHR$(19)" ";
100 SYS49152:PRINTCHR$(17)CHR$(157)MID$(B$,I,1)
110 FORJ=6TO0STEP-1
```

```
120 POKE53270,(PEEK(53270)AND248)+J
130 NEXT
140 NEXT
150 GOTO70
1000 DATA120,173,18,208,201,80,208,249
1010 DATA173,17,208,41,128,208,242,169
1020 DATA20,32,202,241,173,22,208,9
1030 DATA7,141,22,208,88,96
```

Le modifiche apportate a GRAF23, per ottenere GRAF25, sono indicate nel commento.

COMMENTO A GRAF25.
Linea 5: carica in memoria da 49152 (C000H) la routine assembler GRAF24.
Linea 100: sostituito ai comandi che cancellano e spostano le scritte a destra, il comando che salta alla routine in linguaggio macchina.
Linee 1000-1030: contengono i dati relativi alla routine GRAF24.

Per uno scorrimento fine verticale le cose si complicano ulteriormente: infatti, se chiediamo uno scorrimento fine mentre il registro di linea è in una posizione compresa tra l'inizio e la fine dello schermo, otteniamo uno spiacevole sfarfallare dello schermo e lo stesso vale, a maggior ragione, quando chiediamo un'operazione del tipo 7 passi indietro più 8 avanti. Per risolvere questi problemi bisogna quindi sincronizzare gli scorrimenti fini con il registro di linea e compiere lo scorrimento di 8 passi in maniera rapidissima. Purtroppo neppure una routine in linguaggio macchina riesce ad essere così veloce da far scorrere lo schermo mentre il registro di linea è fuori dallo schermo. Abbiamo quindi fatto una routine in linguaggio macchina, la GRAF26, che dispone di 2 pagine video: quando viene visualizzata la prima il BASIC provvede ad aggiungere una linea nuova nella parte nascosta della pagina video (usiamo 24 righe) e la routine in linguaggio macchina trasferisce nella seconda le scritte già posizionate una linea più in alto (questa operazione può essere fatta in maniera relativamente lenta poichè la seconda pagina non viene visualizzata). Al fatidico momento degli 8 passi avanti e dei 7 indietro non facciamo altro che cambiare pagina video e compiere solo 7 passi indietro. Essendo unica la mappa del colore non è stato possibile alternarne due e perciò tutte le scritte che si vedono scorrere saranno dello stesso colore.
Guardiamo prima la routine assembler GRAF26 e poi il programma GRAF27.

COMMENTO A GRAF26.
Dall'indirizzo 49152 (C000H) ha il compito di fare scorrimenti fini sincronizzati col registro di linea e, se è il caso (cioè ogni 8 passi), cambiare la mappa video. Dall'indirizzo 49205 (C035H) inizia la routine che trasferisce una pagina nell'altra.

GRAF26

```
.C000 78        SEI
  ;maschera l'interrupt
,C001 AD 12 D0 LDA $D012
,C004 C9 FF     CMP #$FF
,C006 D0 F9     BNE $C001
,C008 AD 11 D0 LDA $D011
,C00B 29 80     AND #$80
,C00D D0 F2     BNE $C001
  ;attende che il VIC II stia disegnando
  ;la parte non visibile dello schermo
,C00F A5 02     LDA $02
,C011 D0 14     BNE $C027
  ;se non deve cambiare mappa video salta
,C013 AD 18 D0 LDA $D018
,C016 49 10     EOR #$10
,C018 8D 18 D0 STA $D018
  ;cambia mappa video nel VIC II
,C01B AD 88 02 LDA $0288
,C01E 49 04     EOR #$04
,C020 8D 88 02 STA $0288
  ;avvisa il sistema operativo
,C023 A9 08     LDA #$08
,C025 85 02     STA $02
  ;ricarica il contatore
,C027 C6 02     DEC $02
  ;decrementa il contatore (qui anche
  ;quando non cambia mappa video
,C029 AD 11 D0 LDA $D011
,C02C 29 F0     AND #$F0
,C02E 18        CLC
,C02F 65 02     ADC $02
,C031 8D 11 D0 STA $D011
```

```
   ;Pone nei 3 bit meno significativi del
   ;registro 17 del VIC II il contenuto del
   ;contatore Pone il video a 24 linee
 ,C034 60        RTS
   ;torna al basic
 ,C035 AD 88 02 LDA $0288
 ,C038 C9 3C     CMP #$3C
   ;guarda qual'e' la Pagina video corrente
 ,C03A D0 1D     BNE $C059
   ;se e' la Prima ($0380-03C0) salta
 ,C03C A9 3C     LDA #$3C
 ,C03E 85 FF     STA $FF
 ,C040 A9 28     LDA #$28
 ,C042 85 FE     STA $FE
 ,C044 A9 38     LDA #$38
 ,C046 85 FD     STA $FD
 ,C048 A9 00     LDA #$00
 ,C04A 85 FC     STA $FC
 ,C04C A9 3B     LDA #$3B
 ,C04E 8D 9E C0 STA $C09E
 ,C051 A9 C0     LDA #$C0
 ,C053 8D 9D C0 STA $C09D
   ;sistema i Puntatori Per trasferire
   ;la seconda Pagina
 ,C056 18        CLC
 ,C057 90 1A     BCC $C073
   ;salta sempre (abbiamo Preferito fare in
   ;questo modo Per rendere il Programma
   ;rilocabile con maggior facilita'
 ,C059 A9 38     LDA #$38
 ,C05B 85 FF     STA $FF
 ,C05D A9 28     LDA #$28
 ,C05F 85 FE     STA $FE
 ,C061 A9 3C     LDA #$3C
 ,C063 85 FD     STA $FD
 ,C065 A9 00     LDA #$00
 ,C067 85 FC     STA $FC
 ,C069 A9 3F     LDA #$3F
 ,C06B 8D 9E C0 STA $C09E
 ,C06E A9 C0     LDA #$C0
 ,C070 8D 9D C0 STA $C09D
```

```
  ;sistema i Puntatori Per trasferire
  ;la Prima Pagina
,C073 A0 00     LDY #$00
,C075 B1 FE     LDA ($FE),Y
,C077 91 FC     STA ($FC),Y
,C079 E6 FC     INC $FC
,C07B D0 02     BNE $C07F
,C07D E6 FD     INC $FD
,C07F E6 FE     INC $FE
,C081 D0 02     BNE $C085
,C083 E6 FF     INC $FF
,C085 A5 FD     LDA $FD
,C087 CD 9E C0  CMP $C09E
,C08A D0 E9     BNE $C075
,C08C A5 FC     LDA $FC
,C08E CD 9D C0  CMP $C09D
,C091 D0 E2     BNE $C075
  ;trasferisce la Pagina video
,C093 A0 27     LDY #$27
  ;$27=40 numero di caratteri in una linea
,C095 A9 20     LDA #$20
  ;$20=32 codice ASCII dello sPazio
,C097 91 FC     STA ($FC),Y
,C099 88        DEY
,C09A 10 FB     BPL $C097
  ;cancella l'ultima linea
,C09C 60        RTS
  ;torna al basic
```

```
1 REM GRAF27
10 POKE56,56:CLR
20 FORI=1TO40:SP$=SP$+CHR$(32):NEXT
30 G$=CHR$(19):FORI=1TO24:G$=G$+CHR$(17):NEXT
40 FORI=49152TO49308:READA:POKEI,A:NEXT
50 READN
60 DIMLN$(N-1)
70 FORI=0TON-1
80 READLN$(I)
90 IFLEN(LN$(I))>40THEN290
100 LN$(I)=LEFT$(SP$,(40-LEN(LN$(I)))/2)+LN$(I)
```

```
110 NEXT
120 POKE53281,14
130 POKE648,56:PRINTCHR$(147)
140 POKE648,60:PRINTCHR$(147)
150 POKE53280,0:POKE53281,0
160 POKE53272,(PEEK(53272)AND15)+240
170 POKE2,7
180 FORI=0TON-1
190 PRINTG$;LN$(I);
200 FORK=0TO20:NEXT
210 SYS49205
220 SYS49152
230 FORJ=1TO7
240 FORK=1TO60:NEXT
250 SYS49152
260 NEXTJ
270 NEXTI
280 GOTO180
290 PRINT"LINEA "I+1" TROPPO LUNGA":END
10000 DATA120,173,018,208,201,255,208,249
10010 DATA173,017,208,041,128,208,242,165
10020 DATA002,208,020,173,024,208,073,016
10030 DATA141,024,208,173,136,002,073,004
10040 DATA141,136,002,169,008,133,002,198
10050 DATA002,173,017,208,041,240,024,101
10060 DATA002,141,017,208,096,173,136,002
10070 DATA201,060,208,029,169,060,133,255
10080 DATA169,040,133,254,169,056,133,253
10090 DATA169,000,133,252,169,059,141,158
10100 DATA192,169,192,141,157,192,024,144
10110 DATA026,169,056,133,255,169,040,133
10120 DATA254,169,060,133,253,169,000,133
10130 DATA252,169,063,141,158,192,169,192
10140 DATA141,157,192,160,000,177,254,145
10150 DATA252,230,252,208,002,230,253,230
10160 DATA254,208,002,230,255,165,253,205
10170 DATA158,192,208,233,165,252,205,157
10180 DATA192,208,226,160,039,169,032,145
10190 DATA252,136,016,251,096
20000 DATA10
```

```
20010 DATA"**********************"
20020 DATA"*                    *"
20030 DATA"*   COMMODORE 64     *"
20040 DATA"*                    *"
20050 DATA"* SMOOTH SCROLLING *"
20060 DATA"*                    *"
20070 DATA"**********************"
20080 DATA"","",""
```

COMMMENTO A GRAF27.
Linea 10: pone la fine della memoria a 14336 (3800H) dove inizierà la prima pagina video, la seconda inizierà in 15360 (3C00H).
Linea 20: riempie la variabile SP$ con 40 spazi.
Linea 30: la variabile contiene HOME e 24 CURSORE GIU'; quando verrà stampata il cursore si posizionerà all'inizio dell' ultima linea.
Linea 40: carica in memoria la routine in linguaggio macchina.
Linea 50: legge il numero di stringhe che dovrà visualizzare.
Linea 60: dimensiona un vettore di stringhe che conterrà le N stringhe richieste.
Linee 70-110: riempono la matrice controllando che le stringhe non siano più lunghe di 40 caratteri e centrandole.
Linea 120: schermo blù chiaro.
Linea 130: avvisa il sistema operativo che la pagina video è ora in 14336 (3800H) (vedi Paragrafo 2.3.2) e pulisce lo schermo. Pulendo lo schermo mentre questo è blu chiaro il COMMODORE 64 riempie automaticamente la mappa del colore di blù chiaro in modo che, portando lo schermo a nero, basterà mettere il codice del carattere nella mappa video e questo comparirà di colore blu chiaro.
Linea 140: avvisa il sistema operativo che la pagina video è ora in 15360 (3C00H) e pulisce lo schermo.
Linea 150: schermo e bordo neri.
Linea 160: sposta, nel VIC II, la pagina video a 15360 (3C00H).
Linea 170: inizializza il contatore della routine assembler.
Linea 180: inizializza un ciclo.
Linea 190: scrive sull'ultima linea la i-esima stringa del vettore.
Linea 200: attende perchè lo scorrimento non sia troppo veloce.
Linea 210: salta alla routine che trasferisce la pagina video.
Linea 220: salta alla routine che fa scorrere lo schermo.
Linee 230-260: per 7 volte fa uno scorrimento e attende (il ritardo qui è maggiore per compensare il tempo impiegato dalla routine di trasferimento della mappa video).

Linea 270: chiude il ciclo iniziato nella linea 180.
Linea 280: ricomincia fino a che il programma non viene fermato.
Linee 10000-10190: contengono i dati relativi alla routine in linguaggio macchina GRAF26.
Linea 20000: numero delle stringhe che si vogliono visualizzare.
Linee 20010-20080: stringhe che si vogliono visualizzare.

CAPITOLO 3

GLI SPRITE

3.1 COSA E' UNO SPRITE

Probabilmente già sai dal manuale di istruzioni cosa sono gli SPRITE; sono dei rettangoli che puoi spostare molto facilmente, e che puoi disegnare, colorare, cambiare in maniera molto veloce e molto semplice.
In inglese SPRITE significa SPIRITO, e infatti gli sprite sono una sorta di folletti che appaiono e scompaiono con una facilità eccezionale. La loro invenzione è stata fatta dai costruttori di videogiochi, dove si sono rivelati eccezionalmente comodi ai programmatori nel realizzare le animazioni. Sono dei dispositivi abbastanza difficili da realizzare, e il COMMODORE 64 è uno dei primi calcolatori di costo modesto che li possiede.
Ma passiamo a vedere come sono fatti.
Abbiamo detto che sono dei rettangoli: la loro dimensione infatti è di 24 X 21 punti (504 punti): 24 orizzontali e 21 verticali. Possiamo quindi vedere lo sprite come una matrice di 21 X 24 punti che si può spostare attraverso lo schermo. In termine di byte la zona di memoria dedicata a uno sprite è formata da 63 byte, organizzati in 21 righe di 3 byte ciascuna.
Gli sprite si definiscono per punti in modo simile ai caratteri programmabili. Nella situazione normale, ad alta risoluzione, i punti corrispondenti ai bit 1 appaiono del colore dell'inchiostro, mentre quelli corrispondenti ai bit 0 risultano trasparenti, cioè lasciano vedere il colore di quello che sta sotto, sfondo o altro. Le immagini degli sprite possono essere memorizzate solo in blocchi di byte che inizino in un indirizzo multiplo di 64; il primo byte si riferisce agli 8 punti dell'angolo in alto a sinistra, sulla prima riga. E' possibile definire contemporaneamente 8 sprite, numerati da 0 a 7, e mostrare sul video selettivamente quello, o quelli, che di volta in volta si vuole. Per disegnare uno sprite basta prendere un foglio di carta a quadretti, riquadrare un rettangolo avente 24 quadretti di base e 21 di altezza e disegnare la sagoma voluta annerendo i quadretti. Per passare ai numeri da assegnare ai 63 byte che servono a definire il disegno in memoria, si devono considerare le 3 terne di 8 quadretti di ogni riga e trovare i numeri corrispondenti; si deve scrivere 0 per i quadretti non anneriti e 1 per quelli anneriti. La Figura 3.1 riporta uno schema di cosa serve per preparare uno sprite.

```
BYTE 24   BYTE 25   BYTE 26
|_|_|_|_|_|_|_|_| |_|_|_|_|_|_|_|_| |_|_|_|_|_|_|_|_|
BYTE 27   BYTE 28   BYTE 29
|_|_|_|_|_|_|_|_| |_|_|_|_|_|_|_|_| |_|_|_|_|_|_|_|_|
BYTE 30   BYTE 31   BYTE 32
|_|_|_|_|_|_|_|_| |_|_|_|_|_|_|_|_| |_|_|_|_|_|_|_|_|
BYTE 33   BYTE 34   BYTE 35
|_|_|_|_|_|_|_|_| |_|_|_|_|_|_|_|_| |_|_|_|_|_|_|_|_|
BYTE 36   BYTE 37   BYTE 38
|_|_|_|_|_|_|_|_| |_|_|_|_|_|_|_|_| |_|_|_|_|_|_|_|_|
BYTE 39   BYTE 40   BYTE 41
|_|_|_|_|_|_|_|_| |_|_|_|_|_|_|_|_| |_|_|_|_|_|_|_|_|
BYTE 42   BYTE 43   BYTE 44
|_|_|_|_|_|_|_|_| |_|_|_|_|_|_|_|_| |_|_|_|_|_|_|_|_|
BYTE 45   BYTE 46   BYTE 47
|_|_|_|_|_|_|_|_| |_|_|_|_|_|_|_|_| |_|_|_|_|_|_|_|_|
BYTE 48   BYTE 49   BYTE 50
|_|_|_|_|_|_|_|_| |_|_|_|_|_|_|_|_| |_|_|_|_|_|_|_|_|
BYTE 51   BYTE 52   BYTE 53
|_|_|_|_|_|_|_|_| |_|_|_|_|_|_|_|_| |_|_|_|_|_|_|_|_|
BYTE 54   BYTE 55   BYTE 56
|_|_|_|_|_|_|_|_| |_|_|_|_|_|_|_|_| |_|_|_|_|_|_|_|_|
BYTE 57   BYTE 58   BYTE 59
|_|_|_|_|_|_|_|_| |_|_|_|_|_|_|_|_| |_|_|_|_|_|_|_|_|
BYTE 60   BYTE 61   BYTE 62
|_|_|_|_|_|_|_|_| |_|_|_|_|_|_|_|_| |_|_|_|_|_|_|_|_|


IL DISEGNO VIENE TRACCIATO USANDO UN
CARATTERE QUALUNQUE PER ANNERIRE
LE CASELLINE DELLA MATRICE 24 X 21

I BYTE VENGONO RIEMPITI CON BIT
1 IN CORRISPONDENZA DEI CARATTERI
E COMPLETANDO CON BIT 0
```

Figura 3.1 Schemi per la preparazione degli SPRITE ad ALTA RISOLUZIONE

Ora, dopo aver ricordato le prime informazioni sugli sprite, proviamo il programma SPRITE1, che mostra uno sprite molto elementare sul video. Nel commento ai programmi esempio approfondiremo la conoscenza degli sprite. Ti raccomandiamo di non cambiare i numeri di linea: infatti in seguito aggiungeremo delle altre linee e costruiremo un semplice gioco, poco alla volta.

```
1 REM SPRITE1
10 REM SPRITE NERO FERMO
50 REM INDIRIZZO VIC (VIDEO INTERFACE CHIP)
51 REM (CIRCUITO GESTIONE VIDEO)
53 VI=53248
100 FORI=896 TO 958
110 POKEI,255
120 NEXTI:REM PONE 255 NELLE CELLE 896-959
130 PRINTCHR$(147)
140 REM IND. BLOCCO DI DATI/64 (896/64=14)
145 POKE2040,14
150 POKEVI,170:REM POSIZIONE ORIZZONTALE
160 POKEVI+1,120:REM POSIZIONE VERTICALE
170 POKEVI+39,0:REM COLORE NERO
180 POKEVI+21,1:REM VISUALIZZA SPRITE NUM. 0
190 PRINT"PER TOGLIERE LO SPRITE PREMI:"
195 PRINT"STOP+RESTORE"
```

Avviando il programma SPRITE1 vedi un rettangolo nero in una certa posizione dello schermo, e il cursore che lampeggia. Prova a portare il cursore vicino allo sprite usando le frecce di movimento, e a farcelo passare sopra. Ti stupisci? Il cursore non riesce a sovrapporsi allo sprite, ma gli passa sotto. Lo sprite infatti, in questo momento, ha una priorità più alta di quella dello sfondo, e quindi vi si sovrappone.

Prova a richiedere il listato del programma; vedrai che lo sfondo si sposta come sempre verso l'alto (scrolling), ma lo sprite non si muove. Anche premendo SHIFT-CLR/HOME lo sprite rimane lì; è proprio sovrapposto allo schermo. Per mandarlo via puoi premere RUN/STOP-RESTORE.

Gli sprite non si cancellano, ma si spostano.

Lo sprite numero 0 del nostro programma è memorizzato nei byte da 896 a 958. Prova a eseguire in immediato qualche POKE in uno o più dei 63 byte che lo definiscono, portandoli a 0; vedrai crearsi dei buchi nel rettangolo, attraverso i quali compare lo sfondo.

COMMENTO A SPRITE1.
Linee 50-53: gli sprites sono gestiti da un circuito integrato, il VIC II, dotato di diversi registri, che sono adiacenti e partono dal byte 53248. Assegnamo appunto a VI questo valore.
Linee 100-120: poniamo 255 nei byte che vanno da 896 a 958; sono 63 byte nei quali descriviamo lo sprite come un rettangolo pieno. Infatti 255 in binario si rappresenta come 11111111, e, in conseguenza, tutti i punti dello sprite sono ACCESI.
Linea 130: cancella lo schermo.
Linea 140: dato che abbiamo deciso di porre i dati che descrivono l'immagine dello sprite nelle locazioni che vanno da 896 in su, informiamo il VIC II che i dati relativi allo sprite partono dalla locazione 896. Di fatto le immagini degli sprites si possono porre solo in locazioni che partono da un multiplo di 64, e per questo abbiamo scelto proprio 896, che equivale a 64*14. Nel byte 2040, che è il puntatore allo sprite numero 0, poniamo quindi 14.
Linea 150: il primo registro interno del VIC II indica la X, coordinata orizzontale, dello sprite numero 0; noi poniamo lo sprite a 170 punti di distanza dall'asse delle Y, in un sistema particolare di coordinate, usato per gli sprite.
Linea 160: poniamo nel secondo registro il valore della coordinata verticale del nostro sprite (l'origine è in alto a sinistra, ed è fuori dal rettangolo visibile), posizionandolo a 120 punti di distanza dall'asse delle X.
Linea 170: il byte 53287 (40-esimo registro del VIC II) indica di che colore deve essere lo sprite numero 0 (nero in questo caso).
Linea 180: il byte 53269 (22-esimo registro del VIC II) è molto importante; esso indica quali sprites devono essere accesi (cioè mostrati sul video), e quali spenti. Il bit meno significativo (quello di posizione 0) comanda l'accensione dello sprite 0.
Linee 190-195: stampa sullo schermo il messaggio.

Vediamo ora come è organizzato il sistema di coordinate usato dal COMMODORE 64 per gli sprite. Abbiamo a disposizione un quadrante di coordinate, nel quale l'asse X si trova in alto, fuori dalla parte visibile, orientato verso destra, e l'asse delle Y si trova a sinistra, fuori dalla parte visibile, orientato verso il basso. Il video può trovarsi nelle due situazioni di 40 caratteri per 25 linee o di 38 caratteri per 24 linee; gli schemi relativi sono riportati nelle Figure 3.6 e 3.7. In ambedue le figure abbiamo segnato le coordinate dei quattro punti estremi della zona visibile dello schermo.

Affinchè uno sprite sia completamente visibile, è necessario che le coordinate X e Y del suo angolo superiore sinistro siano tali che l'intero rettangolo cada entro il video. Per uno sprite di dimensioni normali, con video 40 X 25, i limiti per X e Y sono:

24 < =X < =320
50 < =Y < =229

infatti 320+24=344 e 229+21=250.

Con video 38 X 24, i limiti diventano:

31 < =X < =311

54 < =Y < =225.

Gli sprite possono essere espansi nelle due direzioni raddoppiando il numero dei punti. In questo caso i limiti per le coordinate sono:

video 40 X 25

24 < =X < =296

50 < =Y < =208

video 38 X 24

31 < =X < =287

50 < =Y < =204.

Le coordinate devono essere considerate come CIRCOLARI; ciò significa che uno sprite espanso, situato come indicato nella Figura 3.8, ha per l'angolo superiore sinistro le seguenti coordinate:

X=488, infatti 488=512-24,

Y=208, infatti 208=250-42.

3.2 MOVIMENTO DI UNO SPRITE

Ora che hai visto uno sprite, probabilmente non sei soddisfatto dell'uso che ne abbiamo fatto; infatti un rettangolo nero si poteva disegnare in BASIC, con poche istruzioni POKE o addirittura con poche istruzioni PRINT. Il programma SPRITE1 non evidenzia assolutamente quello che è il maggior pregio degli sprite: la mobilità. Aggiungiamo al programma SPRITE1 le linee 55-57, 60, 150, 190-570 e cancelliamo la 160 e la 195; queste modifiche servono per far muovere il rettangolino nero atttraverso lo schermo. Per comodità riportiamo per intero il listato del programma SPRITE2, ottenuto con le modifiche.

```
1 REM SPRITE2
10 REM SPRITE CHE SI MUOVE
50 REM INDIRIZZO VIC (VIDEO INTERFACE CHIP)
51 REM (CIRCUITO GESTIONE VIDEO)
53 VI=53248
55 REM INDIRIZZO SID (SOUND INTERFACE DEVICE)
57 REM (CIRCUITO GESTIONE SUONO)
59 SI=54272
60 DX=4:DY=3:X=24:Y=50
100 FORI=896 TO 958
110 POKEI,255
```

```
120 NEXTI:REM PONE 255 NELLE CELLE 896-959
130 PRINTCHR$(147)
140 REM IND. BLOCCO DI DATI/64 (896/64=14)
145 POKE2040,14
150 POKEVI,24:POKEVI+1,50
170 POKEVI+39,0:REM COLORE NERO
180 POKEVI+21,1:REM VISUALIZZA SPRITE NUM. 0
190 HX=X/256:LX=XAND255
200 REM POSIZIONE SPRITE
205 POKEVI,LX:POKEVI+16,HX:POKEVI+1,Y
210 X=X+DX:IFX>320 OR X<24THENDX=-DX:GOSUB500
220 Y=Y+DY:IFY>229 OR Y<50 THENDY=-DY:GOSUB 500
230 GOTO190
500 REM FA BIP
510 POKESI+6,240:REM SOUSTAIN
520 POKESI+24,15:REM VOLUME
530 POKESI+1,80:REM MSB FREQUENZA
540 POKESI+4,17:REM ONDA TRIANGOLARE+ START
550 FORR=1TO3:NEXT:REM ATTESA
560 POKESI+4,16:REM ONDA TRIANGOLARE+STOP
570 RETURN
```

Il programma SPRITE2 ti mostra lo stesso rettangolo di prima, che però si muove attraverso lo schermo, e che rimbalza quando arriva ai bordi, producendo anche un piccolo suono (se il volume del televisore lo consente). Commentiamo solo le linee aggiunte.

COMMENTO A SPRITE2.
Linee 55-59: chiamiamo SI l'indirizzo del primo dei registri interni del SID, che è il circuito di gestione del suono. L'indirizzo di questo byte è 54272.
Linea 60: lo sprite si sposta se cambiamo nel tempo le sue coordinate. Noi aggiungiamo ogni volta alla X (posizione orizzontale) la variabile DX e alla Y la variabile DY. X e Y sono poste inizialmente a 24 e a 50, cioè in alto a sinistra. Il rapporto DY/DX determina l'angolo della traiettoria dello sprite. Diminuendo entrambi i valori degli incrementi delle coordinate, lo sprite si muove più lentamente, e più accuratamente (il passo è più piccolo, e lo sprite si muove poco alla volta). Aumentando DX e DY lo sprite si muove più in fretta, ma fa dei salti passando da una posizione all'altra. Se il BASIC fosse più veloce, si potrebbe lasciare piccolo il passo, e il movimento sarebbe veloce e continuo; comunque pensiamo che non sia necessario aggiungere a questo programma esempio delle routine in linguaggio

macchina, che lo renderebbero molto più veloce, dato che si riesce ad ottenere un buon risultato anche alla velocità del BASIC. Vorremmo comunque sottolineare il fatto che, in generale, il BASIC poco si adatta alla gestione della grafica, in quanto in tale campo la velocità di elaborazione è veramente essenziale per ottenere effetti brillanti.
Linea 150: assegnamo le coordinate iniziali.
Linea 190: assegnamo ora dei valori a due nuove variabili, HX e LX, che sono il byte alto e il byte basso della coordinata X. Vediamo perchè è necessario operare così. La variabile X può infatti superare il valore di 255, il massimo che possiamo mettere nel registro che determina la posizione orizzontale. I punti orizzontali di cui è composto lo schermo sono infatti 320, e noi vogliamo poter portare il nostro sprite anche oltre il 255-esimo punto. Per questa ragione esiste un registro, nel VIC II, che contiene il nono bit, il più significativo, della coordinata orizzontale di ciascuno sprite. Questo registro è il 17-esimo (numero 16, VI+16), e i suoi 8 bit sono i più significativi di ciascuna posizione X. Per lo sprite 0, di cui ci stiamo occupando, la posizione X è determinata dai seguenti 9 bit: bit 0 registro numero 16 + registro numero 0. Per lo sprite 1, invece: bit 1 registro numero 16 + registro numero 2; e così analogamente per gli altri 6 sprites.
A titolo di esempio consideriamo il seguente caso:
registro numero 16 = 00000001 binario
registro numero 0 = 00000100 binario
La posizione X dello sprite 0 in questo caso è quindi 100000100 binario, cioè 256+4, cioè 260 decimale.

Ovviamente la ragione di tali complicazioni sta nel fatto che la CPU del COMMODORE 64, la 6510, ha solo 8 bit sul bus dati, cioè ha solo 8 fili che la collegano con i vari dispositivi esterni (byte di memoria e registri), quindi, per trasmettere una informazione composta di 9 bit, occorre scrivere in due registri diversi.
Linee 200-205: X è stata ora divisa tra due variabili, HX e LX. Poniamo LX nel registro numero 0 del VIC II, e HX nel registro numero 16. LX contiene un numero da 0 a 255, cioè lo stesso numero binario di X, fatta eccezione per il nono bit, che qui è considerato sempre a zero. Se consideriamo l'esempio di prima, dove X valeva 260, LX vale 4, cioè 00000100 binario. HX, invece, tiene conto solo del nono bit, e vale 0 o 1 in accordo col valore del nono bit di X. Nel caso di prima HX valeva 1. Visto che noi visualizziamo sempre solo lo sprite 0, possiamo porre direttamente HX nel registro numero 16, in quanto i 7 bit più significativi, che noi azzeriamo sempre, non sono utilizzati. Se avessimo voluto usare lo sprite 2, anzichè lo sprite 0, anzichè HX avremmo dovuto porre HX*2↑ (numero sprite), cioè HX*4 nel caso dello sprite 2, o HX*8 nel caso dello sprite 3 (2↑3=8), e così via. Poichè 2↑0=1, è evidente che non scriviamo HX*1, in quanto moltiplicare un numero per uno è la stessa cosa che lasciarlo così com'è.
Linea 210: aggiorniamo il valore di X aggiungendo alla stessa il valore di DX. Nota

che DX viene cambiato di segno se lo sprite arriva al bordo, e viene emessa una nota per rendere più autentico l'effetto dell'urto.
Linea 220: viene fatto lo stesso per Y.
Linea 230: si torna alla linea 190, in modo da visualizzare lo sprite nella nuova posizione.
Linea 500: questa è la subroutine che viene chiamata per emettere il BIP che si sente quando lo sprite arriva al bordo.
Linea 510: ponendo 240 nel settimo registro del SID (byte 54278) poniamo SOUSTAIN=15 e DECAY=0. Non scriviamo nessun numero nel registro numero 5 (ATTACK e DECAY), perchè all'accensione vi si trova già 0. Noi non facciamo quindi uso, qui, del modulatore di ampiezza che si trova nel SID. Il funzionamento del SID è abbastanza complesso, puoi andare a leggere il Capitolo 4. Se non hai mai sentito i termini ATTACK, DECAY, SUSTAIN, RELEASE, e non hai ancora visto nessun programma che usa il suono, puoi accettare queste poche linee di programma, rimandando a una fase successiva lo studio della generazione dei suoni.
Linea 520: poniamo il volume al massimo.
Linea 530: i registri 0 e 1 (primo e secondo) determinano la frequenza del suono emesso dal SID. Per essere precisi esiste questa relazione tra il numero posto nei registri della frequenza e la frequenza del suono:
F= (FH*256+FL)*FC/16777216 Hz. Dove:

- F è la frequenza del suono.
- FH è il byte più significativo della frequenza, nel nostro caso il contenuto del registro numero 1.
- FL è il byte meno significativo della frequenza, nel nostro caso il contenuto del registro numero 0, che vale 0 dall'accensione del calcolatore.
- FC è la frequenza del clock principale del calcolatore, quello che temporizza tutte le operazioni. Tale clock può avere due diversi valori, a seconda del modello del COMMODORE 64: il modello europeo ha un clock di 985248.4 Hz. La frequenza del suono che noi generiamo è quindi:

80*256*985248.4/16777216, cioè 1202.70 Hz. Tale frequenza è molto precisa e, se vuoi accordare i tuoi strumenti lo puoi fare usando il calcolatore, in quanto è assai facile calcolare la frequenza del suono emesso dal SID.
Linea 540: poniamo nel quinto registro il numero binario 00010001, avviando il generatore di suono a forma d'onda triangolare.
Linea 550: attendiamo per poco tempo, permettendo alla nota una breve durata.
Linea 560: poniamo nel quinto registro il numero binario 00010000, in modo da arrestare il generatore di suono.
Linea 570: torniamo dalla subroutine.

Ora che questo sprite quadrato si muove come se fosse una pallina, vediamo come fare per dargli anche l'aspetto di una pallina.

Riferendoci allo schema della Figura 3.2, disegnamo la pallina, poi calcoliamo i numeri binari di ogni byte e li trasformiamo in numeri decimali da inserire nel programma, per esempio per mezzo di linee DATA.
La pallina può essere disegnata come indicato nella Figura 3.2.

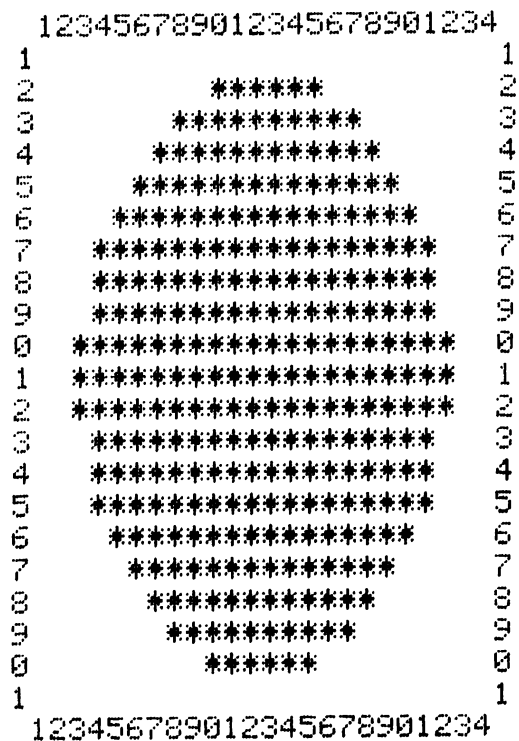



Figura 3.2 Disegno della pallina

Ora dobbiamo trasformare in decimali tutti i 63 numeri binari che ricaviamo dal disegno della nostra pallina.
Per la prima riga è facile: tutti i punti sono spenti, quindi i primi 3 bytes valgono zero.

La seconda riga invece dà luogo a: 00000000 01111110 00000000; i dati ad essa corrispondenti in decimale sono: 0, 126, 0. Noi abbiamo calcolato tutti i 63 dati necessari per descrivere l'immagine della nostra pallina, e li abbiamo aggiunti al programma SPRITE2 sotto forma di linee DATA, ottenendo il programma SPRITE3. Inoltre abbiamo modificato le linee 100-120 nelle 100-110 per caricare i dati della pallina nello sprite numero 0.

```
 1 REM SPRITE3
 10 REM SPRITE CHE SI MUOVE
 50 REM INDIRIZZO VIC (VIDEO INTERFACE CHIP)
 51 REM (CIRCUITO GESTIONE VIDEO)
 53 VI=53248
 55 REM INDIRIZZO SID (SOUND INTERFACE DEVICE)
 57 REM (CIRCUITO GESTIONE SUONO)
 59 SI=54272
 60 DX=4:DY=3:X=24:Y=50
 100 REM PONE DATI PALLINA NEI BYTE 896-959
 110 FORI=896TO958:READA:POKEI,A:NEXTI
 130 PRINTCHR$(147)
 140 REM IND. BLOCCO DI DATI/64 (896/64=14)
 145 POKE2040,14
 150 POKEVI,24:POKEVI+1,50
 170 POKEVI+39,0:REM COLORE NERO
 180 POKEVI+21,1:REM VISUALIZZA SPRITE NUM. 0
 190 HX=X/256:LX=XAND255
 200 REM POSIZIONE SPRITE PALLINA
 205 POKEVI,LX:POKEVI+16,HX:POKEVI+1,Y
 210 X=X+DX:IFX>320 OR X<24THENDX=-DX:GOSUB500
 220 Y=Y+DY:IFY>229 OR Y<50 THENDY=-DY:GOSUB 500
 230 GOTO190
 500 REM FA BIP
 510 POKESI+6,240:REM SOUSTAIN
 520 POKESI+24,15:REM VOLUME
 530 POKESI+1,80:REM MSB FREQUENZA
 540 POKESI+4,17:REM ONDA TRIANGOLARE+ START
 550 FORR=1TO3:NEXT:REM ATTESA
 560 POKESI+4,16:REM ONDA TRIANGOLARE+STOP
 570 RETURN
 1000 DATA0,0,0,0,126,0,1,255,128
```

```
1010 DATA3,255,192,7,255,224,15,255,240
1020 DATA31,255,248,31,255,248,31,255,248
1030 DATA63,255,252,63,255,252,63,255,252
1040 DATA31,255,248,31,255,248,31,255,248
1050 DATA15,255,240,7,255,224,3,255,192
1060 DATA1,255,128,0,126,0,0,0,0
```

Dando il comando RUN vedi che lo sprite che si muove ora ha un'aspetto nuovo; è la pallina che abbiamo disegnato prima.

COMMENTO A SPRITE3.
Linee 100-110: nella zona di memoria che abbiamo dedicato all'immagine dello sprite, poniamo i dati che leggiamo dalle linee DATA.
Linee 1000-1060: contengono i 63 dati che formano la pallina. Ogni linea DATA ha 9 dati, cioè tre righe dello sprite, e in tutto ci sono 7 linee DATA.

Ora, dopo aver fatto muovere sul video degli sprite, ti ricordiamo in cosa differisce l'animazione ottenuta con gli sprite, da quella ottenuta con la grafica normale. Nel caso della grafica normale per produrre animazione si deve:

- porre un oggetto sul video in una posizione A,
- dopo un giusto intervallo di tempo, cancellare l'oggetto dalla posizione A e porlo in una posizione B abbastanza vicina alla posizione A precedente.

Con gli sprite, invece, si visualizza l'oggetto in una posizione, poi lo si sposta, e lo spostamento lo fa sparire dalla precedente posizione.

Ora che questa pallina si muove nello schermo del tuo calcolatore, molto probabilmente ti vengono alla mente i primi videogiochi che sono apparsi nei bar, e anche in versione domestica; giochi come SQUASH, PING PONG, HOCKEY.
E' semplice, ora che c'è la pallina che si muove, aggiungere una racchetta e fare il gioco dello SQUASH.

Disegnamo la nostra racchetta come in Figura 3.3.

```
 123456789012345678901234
1        ********        1
2        ********        2
3        ********        3
4        ********        4
5        ********        5
6        ********        6
7        ********        7
8        ********        8
9        ********        9
0        ********        0
1        ********        1
2        ********        2
3        ********        3
4        ********        4
5        ********        5
6        ********        6
7        ********        7
8        ********        8
9        ********        9
0        ********        0
1        ********        1
 123456789012345678901234
```

Figura 3.3 Disegno della racchetta

I dati di questa racchetta sono davvero semplici: in binario ogni linea è così: 00000000 11111111 00000000, cioè 0, 255, 0 in decimale. Possiamo modificare il programma SPRITE3, ottenendo SPRITE4, con due oggetti sul video, usando anche lo sprite numero 1.

```
1 REM SPRITE4
10 REM SPRITE CHE SI MUOVE
50 REM INDIRIZZO VIC (VIDEO INTERFACE CHIP)
51 REM (CIRCUITO GESTIONE VIDEO)
53 VI=53248
55 REM INDIRIZZO SID (SOUND INTERFACE DEVICE)
```

```
57 REM (CIRCUITO GESTIONE SUONO)
59 SI=54272
60 DX=4:DY=3:X=24:Y=50
70 JA=56320:REM INDIRIZZO PORTA JOYSTICK
100 REM PONE DATI PALLINA NEI BYTE 896-959
110 FORI=896TO958:READA:POKEI,A:NEXTI
125 FORI=960TO1022STEP3:POKEI,0:POKEI+1,255
126 POKEI+2,0:NEXT:REM DATI RACCHETTA
130 PRINTCHR$(147)
140 REM IND. BLOCCO DI DATI/64 (896/64=14)
143 POKE2040,14
145 REM RACCHETTA:IND.BLOCCO DATI/64 (960/64=15)
147 POKE2041,15
150 POKEVI,24:POKEVI+1,50
160 POKEVI+2,255
170 POKEVI+39,0:REM COLORE NERO
171 POKEVI+40,1:REM RACCHETTA COLORE BIANCO
180 POKEVI+21,3:REM VISUALIZZA SPRITE NUM. 0,1
190 HX=X/256:LX=XAND255
200 REM POSIZIONE SPRITE PALLINA
202 POKEVI,LX:POKEVI+16,HX:POKEVI+1,Y
204 IFPEEK(JA)AND1THENAY=AY+4:IFAY>229THENAY=229
205 IFPEEK(JA)AND2THENAY=AY-4:IFAY<50THENAY=50
208 POKEVI+3,AY
210 X=X+DX:IFX>320 OR X<24THENDX=-DX:GOSUB500
220 Y=Y+DY:IFY>229 OR Y<50 THENDY=-DY:GOSUB 500
230 GOTO190
500 REM FA BIP
510 POKESI+6,240:REM SOUSTAIN
520 POKESI+24,15:REM VOLUME
530 POKESI+1,80:REM MSB FREQUENZA
540 POKESI+4,17:REM ONDA TRIANGOLARE+ START
550 FORR=1TO3:NEXT:REM ATTESA
560 POKESI+4,16:REM ONDA TRIANGOLARE+STOP
570 RETURN
1000 DATA0,0,0,0,126,0,1,255,128
1010 DATA3,255,192,7,255,224,15,255,240
1020 DATA31,255,248,31,255,248,31,255,248
1030 DATA63,255,252,63,255,252,63,255,252
1040 DATA31,255,248,31,255,248,31,255,248
1050 DATA15,255,240,7,255,224,3,255,192
1060 DATA1,255,128,0,126,0,0,0,0
```

In SPRITE4 vi sono due oggetti sullo schermo: la pallina, che si muove da sola, e la racchetta, che si muove con il joystick inserito nella porta B. Ti sei sicuramente accorto che la racchetta non colpisce la pallina; non abbiamo ancora aggiunto quella parte di programma, la aggiungeremo alla prossima modifica.

COMMENTO A SPRITE4.
Linea 70: inizializziamo una nuova variabile, JA, che contiene l'indirizzo in cui leggere lo stato del joystick.
Linee 125-126: in questo ciclo poniamo in una zona di memoria i dati che rappresentano l'immagine della racchetta, nei byte che partono dall'indirizzo 960 (15-esimo blocco di 64 bytes).
Linee 145-147: il puntatore dei dati dello sprite 1 viene posto a 15, indicando al VIC II che i dati con l'immagine dello sprite 1 partono dalla locazione di memoria 960.

Linea 160: poichè la racchetta si deve muovere verticalmente, poniamo il valore della sua coordinata X nell'apposito registro del VIC II, cambieremo in seguito solo la sua coordinata Y. Poniamo la X della racchetta a 255.
Linea 171: Indichiamo al VIC II il colore dello sprite numero 1 ponendo il codice nel 41-esimo registro.
Linea 180: a differenza di prima, quando mostravamo un solo sprite, ora ne mostriamo due, e per fare questo poniamo nel registro per l'abilitazione degli sprites il numero 3, che in binario è 00000011; come si vede subito in questo numero vi sono due bit a 1, i bit di posizione 0 e 1. Il VIC II mostrerà adesso lo sprite 0 e lo sprite 1.
Linee 204-205: in queste linee viene letto lo stato del joystick; se si va in basso, si incrementa AY, se si va in alto la si decrementa. La variabile AY contiene la posizione della racchetta: se non vuoi usare il joystick, ma la tastiera, puoi facilmente cambiare queste linee: GET A$: IF A$=... ecc...; se vuoi usare i paddle, adatta tu il programma, sapendo che i valori dei due paddle (numeri da 0 a 255) si trovano negli indirizzi SI+25 e SI+26. Puoi direttamente porre : AY=PEEK(SI+25).
Linea 208: poniamo nel registro 3 del VIC II il valore di AY; il registro 3 indica la posizione verticale dello sprite 1 (quello che usiamo per la racchetta).

E' già stata annunciata la prossima modifica; con l'aggiunta di qualche linea, ora scopriremo l'eventuale collisione tra pallina e racchetta. Aggiungi queste linee:

```
223 CL=PEEK(VI+30)
224 IFCL=3 THEN IF DX > 0 THEN DX=-DX: GOSUB 500
```

per ottenere il programma SPRITE5, del quale non riportiamo il listato, ma lo trovi registrato sulla cassetta.

Avvia l'esecuzione del programma, e ti accorgi che ora la racchetta funziona. Come probabilmente hai già osservato, non abbiamo fatto riferimento alle posizioni di racchetta e pallina; facciamo riferimento solamente a un registro del VIC II, il 31-esimo registro, ed assegnamo alla variabile CL il valore di questo registro. CL sta per COLLISIONE. Il 31-esimo registro del VIC II infatti ci indica se c'è stata collisione tra gli sprite. Se la collisione c'è stata, i bit corrispondenti agli sprite interessati dalla collisione sono a 1. Se quindi si sono scontrati lo sprite 0 (la pallina) con lo sprite 1 (la racchetta) noi troviamo a 1 il bit 0 e il bit 1: troviamo cioè il numero binario 00000011 nel registro collisione sprite-contro sprite (31-esimo registro, il numero 30). Come vedi la pallina si sovrappone alla racchetta al momento dell'impatto; questo dipende dal fatto che la priorità della pallina è superiore a quella della racchetta. Le priorità degli sprite dipendono dal loro numero; lo sprite numero 0 ha priorità 0, la più alta, lo sprite numero 7 ha priorità 7, la più bassa. Nella Figura 3.4 sono indicate le priorità degli sprite.

Linea 223: pone nella variabile CL il valore del registro collisioni sprite-sprite.
Linea 224: se c'è collisione cambia la direzione del moto orizzontale della pallina, ed emette il bip, a patto che la pallina vada da sinistra verso destra (DX > 0).

La priorità tra gli sprite e lo sfondo è controllata dal registro numero 27 (il 28-esimo, di indirizzo 53275); quando il bit relativo a uno sprite è a 0, lo sprite ha priorità maggiore dello sfondo, quando è a 1 lo sfondo prevale sullo sprite. La differenza di priorità fra gli sprite fa sì che un oggetto appaia davanti all'altro e produce un effetto tridimensionale. Se nel programma SPRITE1, dopo il suo arresto, scendi con il cursore e esegui in immediato POKE 53275,1, e dopo sposti il cursore e lo fai passare attraverso lo sprite, questa volta il cursore passa sopra. Infatti per effetto della POKE eseguita, la priorità dello sfondo è diventata superiore a quella dello sprite numero 0.

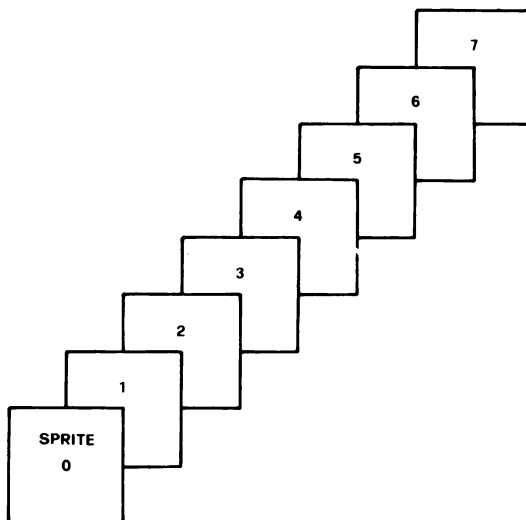



Figura 3.4 Priorità tra gli Sprite

3.3 ALTRI ESEMPI DI ANIMAZIONE

Ora che abbiamo completato il gioco dello SQUASH (SPRITE5), possiamo andare avanti e aggiungere altre modifiche per ottenere il gioco del PING PONG (SPRITE6).
Riportiamo l'intero listato del programma SPRITE6.

```
1 REM SPRITE6
10 REM GIOCO PING PONG
20 REM COLORE QUADRO BLU+SFONDO GRIGIO
25 POKE53280,12:POKE53281,6
50 REM INDIRIZZO VIC (VIDEO INTERFACE CHIP)
51 REM (CIRCUITO GESTIONE VIDEO)
```

```
53 VI=53248
55 REM INDIRIZZO SID (SOUND INTERFACE DEVICE)
57 REM (CIRCUITO GESTIONE SUONO)
59 SI=54272
60 DX=9:DY=6   :X=24:Y=50
70 REM INDIRIZZI PORTE JOYSTICK
75 JA=56320:JB=56321
80 AY=140:BY=AY
100 FORI=896 TO 957
110 READA:POKEI,A
120 NEXTI:REM PONE DATI PALLINA IN MEMORIA
125 FORI=960TO1022STEP3:POKEI,0:POKEI+1,255
126 POKEI+2,0:NEXT:REM DATI RACCHETTE
130 PRINTCHR$(147)
140 REM IND.BLOCCO DATI/64 (896/64=14)
141 POKE2040,14
142 REM RACCHETTA:IND.BLOCCO DATI/64 (960/64=15)
143 POKE2041,15
144 REM IND.BLOCCO DI DATI/64 (958/64=15)
145 POKE2042,15
150 POKEVI,24:POKEVI+1,50:REM POSIZ. INIZIALE
155 POKEVI+2,16:POKEVI+4,72
160 POKEVI+1,120:REM POSIZIONE VERTICALE
170 POKEVI+39,12:REM COLORE PALLINA: GRIGIO
171 POKEVI+40,1:REM COLORE RACCHETTA A: BIANCO
172 POKEVI+41,0:REM COLORE RACCHETTA B: NERO
180 REM VISUALIZZA SPRITES NUM. 0,1,2
185 POKEVI+21,7
190 HX=X/256:LX=XAND255
200 REM POSIZIONE SPRITE
202 POKEVI,LX:POKEVI+16,HX+4:POKEVI+1,Y
204 IFPEEK(JA)AND1THENAY=AY+4:IFAY>229THENAY=229
205 IFPEEK(JA)AND2THENAY=AY-4:IFAY<50THENAY=50
206 IFPEEK(JB)AND1THENBY=BY+4:IFBY>229THENBY=229
207 IFPEEK(JB)AND2THENBY=BY-4:IFBY<50THENBY=50
208 POKEVI+3,AY:POKEVI+5,BY
210 X=X+DX:IFX>345 OR X<1THENDX=-DX:GOSUB600
220 Y=Y+DY:IFY>229 OR Y<50 THENDY=-DY:GOSUB 500
223 CL=PEEK(VI+30)
224 IFCL=3THENIFDX<0THENDX=-DX:GOSUB500
226 IFCL=5THENIFDX>0THENDX=-DX:GOSUB500
```

```
230 GOTO190
500 REM FA BIP
510 POKESI+6,240:REM SOUSTAIN
520 POKESI+24,15:REM VOLUME
530 POKESI+1,80:REM MSB FREQUENZA
540 POKESI+4,17:REM ONDA TRIANGOLARE+START
550 FORR=1TO3:NEXT:REM ATTESA
560 POKESI+4,16:REM ONDA TRIANGOLARE+STOP
570 RETURN
600 REM FA BIIIIP
610 POKESI+4,17:REM ONDA TRIANGOLARE+START
620 FORR=1TO1000:NEXT:REM ATTESA
630 POKESI+4,16:REM ONDA TRIANGOLARE+STOP
640 PB=PB-(DX>0):PA=PA-(DX<0)
650 PRINTCHR$(147)TAB(16)CHR$(5)PACHR$(144)PB
660 RETURN
1000 DATA0,0,0,0,0,0,0,126,0,1,255,128
1010 DATA3,255,192,7,255,224,15,255,240
1020 DATA31,255,248,31,255,248,31,255,248
1030 DATA63,255,252
1040 DATA31,255,248,31,255,248,31,255,248
1050 DATA15,255,240,7,255,224,3,255,192
1060 DATA1,255,128,0,126,0,0,0,0,0,0,0
```

A questo punto nel gioco sono presenti due racchette di colore diverso, comandate dai due joystick, e una pallina; adoperiamo quindi 2 blocchi di dati (pallina e racchetta), ma la racchetta compare in due posizioni diverse con colore diverso. Se vuoi puoi sostituire i joystick con i paddle, ma devi modificare la linea 208 come segue:

208 POKE VI+3, PEEK(SI+25): POKE VI+5, PEEK(SI+26)

Nel commento mettiamo in evidenza solo le novità rispetto ai precedenti programmi esempio.

COMMENTO A SPRITE6.
Linea 20: cambiano i colori dello schermo.
Linea 60: cambiano il passo orizzontale DX e quello verticale DY. Il programma deve ora gestire due racchette, e quindi il tempo che passa tra uno spostamento della pallina e l'altro è più lungo; per questa ragione la pallina compie dei passi più grandi, per non muoversi troppo lentamente.
Linea 70: definiamo anche JB, che è l'indirizzo della porta di I/O dove è collegato il secondo joystick.
Linea 144: inizializziamo il puntatore del blocco dei dati della seconda racchetta.
Linea 155: poniamo nei registri del VIC II le posizioni orizzontali delle racchette.
Linee 160-185: sono spiegate dai commenti di linea.
Linea 190: calcola la parte più significativa e quella meno significativa di X.
Linea 200: posiziona la pallina.
Linee 206-207: legge la posizione del secondo joystick.
Linea 208: aggiorna la posizione delle racchette.
Linea 210: aggiorna la posizione orizzontale della pallina e fa BIIIIP se arriva al bordo senza essere colpita dalla racchetta.
Linea 600: sottoprogramma che genera un suono lungo e aggiorna il punteggio della partita. Funziona come il sottoprogramma in 500.
Linea 640: forse questa linea ti sembra un pò strana. la variabile BP contiene il punteggio del secondo giocatore; essa viene incrementata solo se la pallina andava verso destra (DX > 0), in tale caso l'espressione condizionale (DX > 0) è verificata, e quindi per il calcolatore essa dà luogo a un numero di valore −1. Quando, invece, l'espressione condizionale non viene verificata, questa assume il valore 0.
Linea 650: stampa il punteggio.

Come vedi non sono stati aggiunti concetti nuovi; abbiamo solo vivacizzato un pò il gioco precedente. Puoi provare a cambiare qualche parametro, come la velocità della pallina o delle racchette, per acquistare maggiore dimestichezza con l'argomento dell'animazione con gli sprite.

Ti suggeriamo ora di modificare alcune linee del programma SPRITE6, per ottenere il programma SPRITE7, realizzando così il gioco HOKEY.
Abbiamo aggiunto alcune linee, che sono riportate come MOD-SPRITE6, al programma precedente (SPRITE6), che commentiamo. Non riportiamo il listato completo del programma SPRITE7, che è contenuto sulla cassetta.

```
0 REM MOD-SPRITE6
1 REM SPRITE7
10 REM GIOCO HOKEY
146 REM RACCHETTA:IND.BLOCCO DATI/64 (960/64=15)
147 POKE2043,15
148 REM RACCHETTA:IND.BLOCCO DATI/64 (960/64=15)
149 POKE2044,15
155 REM POSIZ. VERT. RACCHETTE
157 POKEVI+2,16:POKEVI+4,72
159 POKEVI+6,255:POKEVI+8,89
173 POKEVI+42,1:REM COLORE RACCHETTA A: BIANCO
174 POKEVI+43,0:REM COLORE RACCHETTA B: NERO
180 REM VISUALIZZA SPRITES NUM. 0,1,2,3,4
185 POKEVI+21,31
209 POKEVI+7,AY:POKEVI+9,BY
225 IFCL=9THENGOSUB700
227 IFCL=17THENGOSUB800
700 REM COLLISIONE PALLINA-ATTACCANTE A
710 IFDX<0THENDX=-DX:GOSUB500
720 RETURN
800 REM COLLISIONE PALLINA-ATTACCANTE B
810 IFDX>0THENDX=-DX:GOSUB500
820 RETURN
```

COMMENTO A SPRITE7.
Linea 10: cambiato il nome.
Linee 146-149: indirizzo blocco dati seconda racchetta.
Linee 155-159: posizione racchette.
Linea 173: colore prima racchetta.
Linea 174: colore seconda racchetta.
Linee 180-185: visualizza gli sprite.
Linea 209: aggiorna posizione racchetta.
Linee 225 e 227: controlli posizione.
Linee 700-720: collisione pallina-prima racchetta.
Linee 800-820: collisione pallina-seconda racchetta.

Abbiamo preso in esame tutti gli argomenti che è necessario conoscere per produrre animazione con gli sprite. Ora ti presentiamo una serie di programmi, ottenuti uno dall'altro con modifiche successive. In questi programmi ti mostriamo come sia possibile evitare noiosi calcoli per programmare uno sprite.

Cominciamo con il programma SPRITE8; in esso prepariamo la sagoma di un omino in piedi, usando la tecnica di disegnarlo con asterischi e spazi in 21 linee DATA. Provvede il programma a trasformare i contenuti delle linee DATA nei valori da memorizzare nei 63 byte per programmare lo sprite.

```
1 REM SPRITE8
10 REM OMINO FERMO
20 POKE56,32:CLR:POKE53280,0:POKE53281,0
30 DIMD(63)
40 DIMZ(7):FORI=0TO7:Z(I)=2↑I:NEXT
50 VI=53248:REM INDIRIZZO DEL VIC
100 GOSUB2000
110 ND=128:GOSUB2100
130 PRINTCHR$(147)
140 POKE2040,128:REM IND. BLOCCO DI DATI/64
150 POKEVI,170:REM POSIZIONE ORIZZONTALE
160 POKEVI+1,120:REM POSIZIONE VERTICALE
170 POKEVI+39,7:REM COLORE NERO
180 POKEVI+21,1:REM VISUALIZZA SPRITE NUM. 0
190 PRINT"PER TOGLIERE LO SPRITE PREMI:"
195 PRINT"STOP+RESTORE"
200 END
999 REM   123456789012345678901234
1000 DATA"          *****         "
1010 DATA"         **** *         "
1020 DATA"         *******        "
1030 DATA"         ******         "
1040 DATA"          ****          "
1050 DATA"           **           "
1060 DATA"          ****          "
1070 DATA"         **   **        "
1080 DATA"        ***   **        "
1090 DATA"        ***  ***        "
1100 DATA"        ***  ***        "
1110 DATA"        ***  ***        "
1120 DATA"        ****  **        "
1130 DATA"         ******         "
1140 DATA"          ****          "
1150 DATA"           **           "
1160 DATA"           **           "
```

```
1170 DATA"          **         "
1180 DATA"          **         "
1190 DATA"          ****       "
1200 DATA"          ****       "
1210 REM  123456789012345678901234
2000 PRINTCHR$(147);:FORI=0TO20:READA$:PRINTA$
2010 FORJ=0TO2:D(I*3+J)=0:B$=MID$(A$,1+J*8,8)
2020 FORK=1TO8
2025 IFMID$(B$,K,1)="*"THEN2035
2030 NEXTK,J,I:RETURN
2035 D(I*3+J)=D(I*3+J)+Z(8-K):GOTO2030
2100 FORI=0TO63:POKEND*64+I,D(I):NEXT:RETURN
```

COMMENTO A SPRITE8.
Linea 20: pone 32 nel byte di indirizzo 56; questo produce l'effetto di abbassare il top della memoria, lasciando le locazioni di indirizzo superiore a 8192 (32*256=8192) a nostra disposizione, senza che il BASIC possa accedervi.
Linee 30-40: dimensiona il vettore D che conterrà i valori numerici dello sprite e prepara una matrice Z di 8 elementi con le prime 8 potenze del 2 (risulta più veloce accedere ad un elemento di una matrice che calcolare una potenza).
Linee 100-200: prepara i parametri necessari, come indicato dal commento contenuto nelle REM.
Linee 1000-1200: linee DATA con l'immagine grafica dello sprite, disegnata con gli asterischi.
Linee 2030-2035: sottoprogramma che pone in memoria i dati della matrice D a partire dall'indirizzo ND*64. ND=128 significa che i dati sono memorizzati a partire dalla locazione 8192 (128*64=8192), che è la prima di quelle riservate con la linea di programma 20. Mentre vengono calcolati i dati, il programma mostra gli sprite ingranditi sul video.

Con SPRITE8 otteniamo un'omino fermo; le dimensioni dell'omino sono quelle possibili per una sagoma contenuta in un rettangolo di 24 X 21 punti. E' possibile espandere lo sprite nelle due direzioni; si può:

- raddoppiare la larghezza,
- raddoppiare l'altezza,
- raddoppiare sia la larghezza che l'altezza.

Per provare anche questa caratteristica degli sprite, puoi aggiungere al programma SPRITE8 le due linee seguenti:

```
163 POKE VI+23,1:REM ESPANSIONE VERTICALE SPRITE 0
166 POKE VI+29,1:REM ESPANSIONE ORIZZONTALE SPRITE 0
```

e provare il programma. Puoi anche provare cosa succede aggiungendo una sola delle due linee suggerite.

Ora aggiungiamo poche linee a SPRITE8 (nella versione precedente alle modifiche suggerite per ingrandire l'omino) per ottenere il programma SPRITE9, che non listiamo, ma trovi sulla cassetta. In esso il nostro ommino scivola sul video, senza muovere le gambe. Le linee aggiunte sono riportate come MOD-SPRITE8.

```
0 REM MOD-SPRITE8
1 REM SPRITE9
10 REM OMINO CHE SI MUOVE
200 FORX=0TO340
210 HX=X/256:LX=XAND255
220 POKEVI,LX:POKEVI+16,HX
230 NEXT:GOTO200
```

Ora vediamo di migliorare ancora la situazione, cioè facciamo camminare l'omino. Per ottenere il programma SPRITE10, memorizzato sulla cassetta, ma non listato qui per intero, abbiamo aggiunto le linee riportate come MOD-SPRITE9. Queste linee disegnano da 1210 a 1410 un omino in posizione di passo. Le altre linee aggiunte servono per alternare le due immagini (i due sprite) sullo schermo e dare l'impressione del movimento. Le linee 210-215 realizzano il movimento così: la variabile S contiene 0 o 1, a seconda del valore del bit di posizione 2 della variabile X; ogni 4 spostamenti S cambia valore, determinando quale delle due immagini deve essere visualizzata.

```
0 REM MOD-SPRITE9
1 REM SPRITE10
10 REM OMINO CHE CAMMINA
120 GOSUB2000
125 ND=129:GOSUB2100
210 HX=X/256:LX=XAND255:S=-((XAND4)=0)
215 POKE2040,128+S
220 POKEVI,LX+S:POKEVI+16,HX
```

```
1210 DATA"          *****         "
1220 DATA"         **** *         "
1230 DATA"         *******        "
1240 DATA"         ******         "
1250 DATA"          ****          "
1260 DATA"           **           "
1270 DATA"          ****          "
1280 DATA"         **  **         "
1290 DATA"        ***  **         "
1300 DATA"        ****  **        "
1310 DATA"        ****  **        "
1320 DATA"        *****  *        "
1330 DATA"        ******          "
1340 DATA"         ******         "
1350 DATA"          ****          "
1360 DATA"          *****         "
1370 DATA"         **  **         "
1380 DATA"         **  **         "
1390 DATA"         **   **        "
1400 DATA"        ***   ***       "
1410 DATA"        ****  ****      "
```

Per concludere ti presentiamo un semplice gioco per due persone. Esso, realizzato nel programma SPRITE11, deriva dai 3 programmi degli omini, appena presentati. Il gioco consiste nel far mangiare al proprio omino il maggior numero possibile di palline. Per muovere l'omino, ogni giocatore usa un joystick; le palline vengono mangiate urtandole con la pancia. Per bloccare l'avversario si può usare il bottone del fuoco in qualunque momento, l'avversario sviene per un breve periodo, ma tu perdi 5 palline. Le palline vengono distribuite sul video in modo casuale. Nella parte alta del video viene visualizzato il punteggio.

Il programma SPRITE11 non presenta alcuna sostanziale differenza rispetto ai precedenti, per cui ne riportiamo il listato senza commenti. Come vedi abbiamo dovuto disegnare 5 sprite diversi.

```
1 REM SPRITE11
10 REM GIOCHINO DUE OMINI AVVERSARI
110 POKE56,32:CLR:POKE53280,0:POKE53281,0
120 DIMD(63):VI=13*4096:CP=81
125 DIMZ(7):FORI=0TO7:Z(I)=2↑I:NEXT
130 GOSUB1530:ND=128:GOSUB1610
140 GOSUB1530:ND=129:GOSUB1610
150 GOSUB1530:ND=130:GOSUB1610
160 GOSUB1530:ND=131:GOSUB1610
170 GOSUB1530:ND=134:GOSUB1610
180 RESTORE:GOSUB1570:ND=132:GOSUB1610
190 GOSUB1570:ND=133:GOSUB1610
200 POKE2040,130:POKE2041,130:X=128:Y=128
205 W=200:Z=200:POKEVI+21,3
210 POKEVI+39,3:POKEVI+40,2
220 A=RND(-1)
230 POKE53281,7:PRINT"J":POKE53281,0
235 FORI=1TO50:POKE1024+RND(1)*1000,CP:NEXT
240 POKEVI,XAND255:POKEVI+2,WAND255
245 POKEVI+16,((XAND256)/256)OR((WAND256)/128)
250 POKEVI+1,Y:POKEVI+3,Z
260 IFM1THENPOKE2040,134:GOTO280
265 XR=((DIAND2)=0)*(-(XAND4)/4)
266 XS=((DIAND2)=2)*(-(YAND4)/4)
267 XR=XR OR XS OR 128 OR DI
270 POKE2040,XR
280 IFM2THENPOKE2041,134:GOTO300
290 XR=((D2AND2)=0)*(-(WAND4)/4)
293 XS=((D2AND2)=2)*(-(ZAND4)/4)
295 POKE2041,XR OR XS OR 128 OR D2
300 J=NOTPEEK(56320):K=NOTPEEK(56321)
310 IFM1THENM1=M1-1:J=0:K=KAND15
320 IFM2THENM2=M2-1:K=0:J=JAND15
330 IFJAND1THENY=Y-4:Y=YAND255:DI=2
340 IFJAND2THENY=Y+4:Y=YAND255:DI=2
350 IFJAND4THENX=X-4:X=XAND511:DI=4
360 IFJAND8THENX=X+4:X=XAND511:DI=0
370 IFJAND16THENIFP1>4THENP1=P1-5:M2=50
380 IFKAND1THENZ=Z-4:Z=ZAND255:D2=2
390 IFKAND2THENZ=Z+4:Z=ZAND255:D2=2
400 IFKAND4THENW=W-4:W=WAND511:D2=4
```

```
410 IFKAND8THENW=W+4:W=WAND511:D2=0
420 IFKAND16THENIFP2>4THENP2=P2-5:M1=50
430 C=PEEK(VI+31)
440 IFCAND1THEN473
450 IFCAND2THEN476
460 PRINT"◣"P1"▌▌      ▦",P2"▌▌      "
470 GOTO240
473 A=1024+(X-16)/8+INT((Y-40)/8)*40
474 IFPEEK(A)=CPTHENPOKEA,32:P1=P1+1
475 GOTO450
476 A=1024+(W-16)/8+INT((Z-40)/8)*40
477 IFPEEK(A)=CPTHENPOKEA,32:P2=P2+1
478 GOTO460
479 REM  123456789012345678901234
480 DATA"          *****         "
490 DATA"         **** *         "
500 DATA"         *******        "
510 DATA"         ******         "
520 DATA"          ****          "
530 DATA"           **           "
540 DATA"          ****          "
550 DATA"         **  **         "
560 DATA"        ***  **         "
570 DATA"        ***  ***        "
580 DATA"        ***  ***        "
590 DATA"        ***  ***        "
600 DATA"        ****  **        "
610 DATA"         ******         "
620 DATA"          ****          "
630 DATA"           **           "
640 DATA"           **           "
650 DATA"           **           "
660 DATA"           **           "
670 DATA"           ****         "
680 DATA"           ****         "
690 DATA"          *****         "
700 DATA"         **** *         "
710 DATA"         *******        "
720 DATA"         ******         "
730 DATA"          ****          "
740 DATA"           **           "
750 DATA"          ****          "
```

```
 760 DATA"         **  **          "
 770 DATA"        ***  **          "
 780 DATA"        ****  **         "
 790 DATA"        ****  **         "
 800 DATA"        *****  *         "
 810 DATA"        ******           "
 820 DATA"         ******          "
 830 DATA"          ****           "
 840 DATA"          *****          "
 850 DATA"         **  **          "
 860 DATA"         **  **          "
 870 DATA"         **   **         "
 880 DATA"        ***   ***        "
 890 DATA"        ****  ****       "
 900 DATA"          *****          "
 910 DATA"         ******* **      "
 920 DATA"         ******* **      "
 930 DATA"          *****  **      "
 940 DATA"      **   ***   **      "
 950 DATA"      **   ***   **      "
 960 DATA"      *** ***** ***      "
 970 DATA"       * ******* *       "
 980 DATA"        * ***** *        "
 990 DATA"        ** *** **        "
 1000 DATA"        *** * ***        "
 1010 DATA"        **** ****        "
 1020 DATA"         ** * **         "
 1030 DATA"         * *** *         "
 1040 DATA"         * *** *         "
 1050 DATA"          *****          "
 1060 DATA"          ** **          "
 1070 DATA"          ** **          "
 1080 DATA"          **             "
 1090 DATA"          **             "
 1100 DATA"          **             "
 1110 DATA"          *****          "
 1120 DATA"      ** *******         "
 1130 DATA"      ** *******         "
```

```
1140 DATA"      **  *****         "
1150 DATA"      **   ***    **     "
1160 DATA"      **   ***    **     "
1170 DATA"      *** ***** ***     "
1180 DATA"       * ******* *      "
1190 DATA"        * ***** *       "
1200 DATA"        ** *** **       "
1210 DATA"        *** * ***       "
1220 DATA"        **** ****       "
1230 DATA"         ** * **        "
1240 DATA"         * *** *        "
1250 DATA"         * *** *        "
1260 DATA"          *****         "
1270 DATA"          ** **         "
1280 DATA"          ** **         "
1290 DATA"             **         "
1300 DATA"             **         "
1310 DATA"             **         "
1320 DATA"                        "
1330 DATA"                        "
1340 DATA"                        "
1350 DATA"                        "
1360 DATA"                        "
1370 DATA"                        "
1380 DATA"                        "
1390 DATA" ** **                  "
1400 DATA"                        "
1410 DATA"**   **                 "
1420 DATA"   **                   "
1430 DATA"                        "
1440 DATA"                        "
1450 DATA"   *    ***  *          "
1460 DATA" ****  ***   **         "
1470 DATA"* *** ***   ****        "
1480 DATA"*******    *****     ** "
1490 DATA"*******  *******     ** "
1500 DATA"***** ***************** "
1510 DATA" ***   **************** "
1520 DATA"        *************** "
1525 REM  123456789012345678901234
1530 PRINT"⊐";:FORI=0TO20:READA$:PRINTA$
1540 FORJ=0TO2:D(I*3+J)=0:B$=MID$(A$,1+J*8,8)
```

```
1550 FORK=1TO8
1555 IFMID$(B$,K,1)="*"THEND(I*3+J)=D(I*3+J)+Z(8-K)
1560 NEXTK,J,I:RETURN
1570 PRINT"♥";:FORI=0TO20:READA$:PRINTA$
1580 FORJ=0TO2:D(I*3+J)=0:B$=MID$(A$,1+(2-J)*8,8)
1585 FORJ=0TO2:D(I*3+J)=0:B$=MID$(A$,1+(2-J)*8,8)
1590 FORK=1TO8
1595 IFMID$(B$,K,1)="*"THEND(I*3+J)=D(I*3+J)+Z(K-1)
1600 NEXTK,J,I:RETURN
1610 FORI=0TO63:POKEND*64+I,D(I):NEXT:RETURN
```

3.4 SPRITE MULTICOLORE

In tutti i programmi esempio riportati nei precedenti paragrafi abbiamo usato gli sprite nel modo alta risoluzione. Nel capitolo dedicato alla grafica abbiamo visto la differenza tra il modo ALTA RISOLUZIONE e il modo MULTICOLORE. Nel primo ad ogni singolo bit corrisponde un punto, nel secondo un punto corrisponde a due bit, ragione per la quale il punto può essere di uno tra 4 colori diversi, a seconda del valore dei 2 bit che lo definiscono. Per valore 00 abbiamo il colore dello sfondo, per 01 il colore ausiliario numero 0 (registro numero 37), per 10 il colore dello sprite, per 11 il colore ausiliario numero 1 (registro numero 38).

In questo modo i pixel a disposizione per disegnare uno sprite diminuiscono; la matrice diventa 12 X 21. Ogni pixel risulta largo 2 punti ad alta risoluzione e può essere di uno tra 4 colori scelti tra i 16 disponibili.

Nella Figura 3.5 riportiamo gli schemi necessari per disegnare uno sprite multicolore.

```
BYTE 27         BYTE 28         BYTE 29
|_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_|
BYTE 30         BYTE 31         BYTE 32
|_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_|
BYTE 33         BYTE 34         BYTE 35
|_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_|
BYTE 36         BYTE 37         BYTE 38
|_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_|
BYTE 39         BYTE 40         BYTE 41
|_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_|
BYTE 42         BYTE 43         BYTE 44
|_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_|
BYTE 45         BYTE 46         BYTE 47
|_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_|
BYTE 48         BYTE 49         BYTE 50
|_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_|
BYTE 51         BYTE 52         BYTE 53
|_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_|
BYTE 54         BYTE 55         BYTE 56
|_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_|
BYTE 57         BYTE 58         BYTE 59
|_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_|
BYTE 60         BYTE 61         BYTE 62
|_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_| |_|_|


IL DISEGNO VIENE TRACCIATO SCRIVENDO:
1, 2 O 3 (A SECONDA DEL COLORE DESIDERATO)
NELLE CASELLINE DELLA MATRICE 12 X 21

I BYTE VENGONO RIEMPITI CON I BIT
CORRISPONDENTI A 1, 2, 3 (01, 10, 11)
E COMPLETANDO CON BIT 0
```

Figura 3.5 Schemi per la preparazione degli SPRITE MULTICOLORE

Le dimensioni dello sprite non variano dato che il COMMODORE 64 è stato costruito in modo che un punto multicolore ha la larghezza di due punti normali.

La gestione degli sprite multicolori risulta molto semplice; il registro 28 del VIC II (il 29-esimo) dà notizia, con il valore dei suoi 8 bit, di quali sprite devono essere visualizzati in modo multicolore, e di quali in modo normale (alta risoluzione). Nella Figura 3.5 è mostrata la corrispondenza tra i bit della matrice di 63 byte e i punti dello sprite multicolore.
I colori sono 4, di cui uno è quello dello sfondo, uno è quello contenuto nel registro del colore dello sprite, e gli altri due sono i colori contenuti in due registri di colore ausiliario, rispettivamente il numero 37 e il numero 38. Va notato che i due colori contenuti in questi registri sono i colori ausiliari per tutti gli 8 sprite visualizzabili.

Il programma SPRITE12, che ti presentiamo, mostra l'animazione di un atleta che corre. Abbiamo adottato la stessa tecnica del programma SPRITE8 per calcolare i valori da memorizzare per descrivere lo sprite, solo che ora per ANNERIRE ogni punto si usa un numero da 1 a 3, che indica il colore con il quale lo si vuole disegnare. Purtroppo in multicolore l'immagine disegnata non è così immediata come nel modo ad alta risoluzione (risulta più stretta); per ovviare a questo, la routine, che converte i dati grafici in numeri, stampa anche l'immagine dello sprite ingrandita e con i colori originali.

```
1 REM SPRITE12
5 REM ATLETA CHE CORRE
10 POKE56,36:CLR:POKE53281,5
20 DIMD(64),A$(3):VI=53248
30 A$(0)="  "
40 A$(1)=CHR$(18)+CHR$(144)+CHR$(32)+CHR$(146)
50 A$(2)=CHR$(18)+CHR$(158)+CHR$(32)+CHR$(146)
60 A$(3)=CHR$(18)+CHR$(129)+CHR$(32)+CHR$(146)
70 SP$="                                                          "
100 FORNB=144TO150
110 GOSUB9000:GOSUB9100
120 NEXT
130 PRINTCHR$(147):FORI=1TO9:PRINT:NEXT
140 PRINTCHR$(18)CHR$(144)SP$;
150 PRINTCHR$(149);
160 FORI=1TO6:PRINTSP$;:NEXT
170 PRINTCHR$(18)CHR$(144)SP$;
190 POKEVI+1,140:REM POSIZIONE VERTICALE
200 POKEVI+23,1:REM ESPANDE IN ALTEZZA
210 POKEVI+28,1:REM SPRITE MULTICOLOR
219 REM COLORE MULTICOLOR #0 (NERO)
```

```
220 POKEVI+37,0
229 REM COLORE MULTICOLOR #1 (ARANCIO)
230 POKEVI+38,8
239 REM COLORE SPRITE #0 (GIALLO)
240 POKEVI+39,7
250 POKEVI+21,1:REM ACCENDE SPRITE #0
260 FORX=0TO6:FORI=0TO6
270 POKE2040,I+144:Z=(X*7+I)*7
280 POKEVI,ZAND255:POKEVI+16,Z/256
290 FORR=1TO150:NEXT:NEXT:NEXT
300 GOTO260
1000 DATA"            "
1010 DATA"            "
1020 DATA"     11     "
1030 DATA"     133    "
1040 DATA"     133    "
1050 DATA"     33     "
1060 DATA"     23     "
1070 DATA"    232     "
1080 DATA"    2322    "
1090 DATA"    2322    "
1100 DATA"    2322    "
1110 DATA"    2322    "
1120 DATA"    1311    "
1130 DATA"    1111    "
1140 DATA"     111    "
1150 DATA"     33     "
1160 DATA"     33     "
1170 DATA"     22     "
1180 DATA"     221    "
1190 DATA"     111    "
1200 DATA"            "
1210 DATA"            "
1220 DATA"            "
1230 DATA"     11     "
1240 DATA"     133    "
1250 DATA"     133    "
1260 DATA"     33     "
1270 DATA"   223322   "
1280 DATA"  3322223   "
1290 DATA" 33 22223   "
```

```
1300 DATA"  3322223   "
1310 DATA"   332223333"
1320 DATA"    2222    "
1330 DATA"    1111    "
1340 DATA"    11111   "
1350 DATA"   3331133  "
1360 DATA"12233   333 "
1370 DATA"1223     33 "
1380 DATA"1        22 "
1390 DATA"         22 "
1400 DATA"         111"
1410 DATA"            "
1420 DATA"            "
1430 DATA"            "
1440 DATA"     11     "
1450 DATA"     133    "
1460 DATA"     133    "
1470 DATA"     33     "
1480 DATA"    223     "
1490 DATA"    3222    "
1500 DATA"    3222    "
1510 DATA"    3222    "
1520 DATA"    333333  "
1530 DATA"    2222    "
1540 DATA"    1111    "
1550 DATA"    1111    "
1560 DATA"     111    "
1570 DATA"   12333    "
1580 DATA"   12333    "
1590 DATA"   1 22     "
1600 DATA"     22     "
1610 DATA"     111    "
1620 DATA"            "
1630 DATA"            "
1640 DATA"            "
1650 DATA"     11     "
1660 DATA"     133    "
1670 DATA"     133    "
1680 DATA"     33     "
1690 DATA"    222     "
1700 DATA"   32232    "
1710 DATA"  332232    "
```

```
1720 DATA"   32232    "
1730 DATA"    223333  "
1740 DATA"    2222    "
1750 DATA"    1111    "
1760 DATA"    1111    "
1770 DATA"   331111   "
1780 DATA"  333  333  "
1790 DATA" 233    233 "
1800 DATA"222    122  "
1810 DATA"12     12   "
1820 DATA" 11     1   "
1830 DATA"            "
1840 DATA"            "
1850 DATA"            "
1860 DATA"     11     "
1870 DATA"     133    "
1880 DATA"     133    "
1890 DATA"     33     "
1900 DATA"   222222   "
1910 DATA"  3322223   "
1920 DATA" 33 22223   "
1930 DATA"  3322223   "
1940 DATA"   32222333"
1950 DATA"    2222    "
1960 DATA"    1111    "
1970 DATA"    11111   "
1980 DATA"   331 133  "
1990 DATA"12233   333 "
2000 DATA"1223     33 "
2010 DATA"1        22 "
2020 DATA"         22 "
2030 DATA"         111"
2040 DATA"            "
2050 DATA"            "
2060 DATA"            "
2070 DATA"     11     "
2080 DATA"     133    "
2090 DATA"     133    "
2100 DATA"     33     "
2110 DATA"    222     "
2120 DATA"    2232    "
2130 DATA"    2232    "
```

```
2140 DATA"    2232    "
2150 DATA"    223333  "
2160 DATA"    2222    "
2170 DATA"    1111    "
2180 DATA"    1111    "
2190 DATA"     111    "
2200 DATA"   12333    "
2210 DATA"   12333    "
2220 DATA"   1 22     "
2230 DATA"     22     "
2240 DATA"     111    "
2250 DATA"            "
2260 DATA"            "
2270 DATA"            "
2280 DATA"     11     "
2290 DATA"     133    "
2300 DATA"     133    "
2310 DATA"     33     "
2320 DATA"    222     "
2330 DATA"   33222    "
2340 DATA"  332222    "
2350 DATA"   33222    "
2360 DATA"    332233  "
2370 DATA"    2222    "
2380 DATA"    1111    "
2390 DATA"    1111    "
2400 DATA"   331133   "
2410 DATA"  333  333  "
2420 DATA" 233    233 "
2430 DATA"222    122  "
2440 DATA"12     12   "
2450 DATA" 11    1    "
2460 DATA"            "
9000 PRINT"♥":FORI=0TO20:READA$
9010 FORJ=0TO2:FORK=1TO4:B$=MID$(A$,J*4+K,1)
9020 R=VAL(B$):PRINTA$(R);
9030 BY=BY*4+R:NEXT
9040 D(I*3+J)=BY:BY=0:NEXT:PRINT:NEXT
9050 RETURN
9100 FORI=0TO62
9110 POKENB*64+I,D(I)
9120 NEXT:RETURN
```

COMMENTO A SPRITE12.
Linea 10: viene abbassato il top della memoria per riservare locazioni sopra la zona dedicata al BASIC.
Linee 30-60: la matrice A$ contiene 4 elementi; ciascuno di essi serve per stampare uno spazio di colore diverso. A$(0) per spazio nel colore dello sfondo. A$(1) per spazio nero in campo inverso. A$(2) per spazio giallo in campo inverso. A$(3) per spazio arancio in campo inverso.
Linea 70: SP$ contiene 40 spazi.
Linee 100-120: viene richiamata la routine che converte i dati e li memorizza.
Linee 130-170: disegna la pista per l'atleta.
Linee 190-250: prepara i registri per visualizzare lo sprite numero 0 in modo multicolore.
Linee 260-300: sposta l'atleta, cambia il puntatore ai dati (byte 2040) e genera i cicli di attesa necessari per produrre l'animazione.
Linee 1000-2260: linee DATA per preparare le immagini.
Linee 9000-9050: routine che legge 21 linee DATA, le stampa e le converte in 63 numeri che pone nella matrice D.
Linee 9100-9120: routine che pone i dati contenuti nella matrice D in memoria a partire dal byte di indirizzo NB*64. I dati degli sprite vengono memorizzati da 9216 a 9662.

3.5 RIASSUNTO DELLE CARATTERISTICHE DEGLI SPRITE

Il COMMODORE 64 può visualizzare fino a un massimo di 8 sprite.
Gli sprite possono essere di due tipi: alta risoluzione o multicolore.
Nelle Figure 3.1 e 3.5 è mostrata la corrispondenza tra punti e bit in memoria.
Il VIC II contiene diversi registri dedicati alla gestione degli sprite; essi iniziano alla locazione 53248. Gli 8 puntatori alle matrici di descrizione degli sprite si trovano negli 8 byte da 2040 a 2047.
Per visualizzare uno sprite devi:

1) disegnare lo sprite servendoti della griglia riportata nella Figura 3.1 o dei riferimenti indicati nella Figura 3.5;

2) convertire l'immagine in numeri, 63 numeri, come indicato nella Figura 3.1 o nella Figura 3.5;

3) memorizzare i 63 numeri che danno l'immagine dello sprite o nelle locazioni da 832 a 1023 (buffer della cassetta), che possono essere usate solo se si hanno fino a 3 sprite, o sopra il programma BASIC, come indicato nei programmi esempio, ricordando però che non si deve uscire dal banco di memoria utilizzato (inoltre la prima locazione di memorizzazione di uno sprite deve avere indirizzo multiplo di 64);

4) memorizzare negli appropriati registri:
  posizione orizzontale X, registri pari da 0 a 14,
  posizione verticale Y, registri dispari da 1 a 15,
  bit più significativo della posizione X degli 8 sprite, registro 16,
  abilitazione degli sprite, registro 21,
  espansione verticale, registro 23,
  espansione orizzontale, registro 29,
  priorità sullo sfondo, registro 27,
  selezione sprite multicolore, registro 28,
  colore ausiliario numero 0, registro 37,
  colore ausiliario numero 1, registro 38,
  colore dello sprite, registri da 39 a 46;

5) utilizzare, se è il caso i due registri delle collisioni: il 30 per le collisioni tra sprite e il 31 per le collisioni tra sprite e sfondo.

Il movimento dello sprite si ottiene cambiandone la posizione dopo un opportuno intervallo di tempo.

Nei registri usati per tutti gli otto sprite, il bit di posizione 0 è dedicato allo sprite numero 0, il bit di posizione 7 allo sprite numero 7, e così via. Nell'Appendice A sono elencati i registri del VIC II.

I puntatori ai dati descrittivi degli sprite si trovano nei byte di indirizzo da 2040 a 2047, solo quando è attivo il banco numero 0 di memoria per il VIC II e la pagina video è posta all'indirizzo 1024 (0400H). In realtà il VIC II usa come puntatori ai dati degli sprite gli 8 byte che si trovano da VM + 1016 a VM + 1023 (VM = indirizzo primo byte memoria video; vedi Paragrafo 2.3.2). Se si cambia banco, cioè se si modificano i bit di posizione 0 e 1 del registro di indirizzo 56576, allora gli indirizzi dei puntatori sono modificati, sommando ai precedenti indirizzi il numero del banco moltiplicato per 16384. Analogamente gli indirizzi posti nei puntatori devono corrispondere alle locazioni dove sono stati memorizzati gli sprite nel banco di memoria attivo per il VIC II. Questi argomenti sono stati ampiamente trattati nel Capitolo 2.
Per esempio: il byte di indirizzo 56576 contiene 10 nei due bit meno significativi, cioè seleziona per il VIC II il banco numero 1, che corrisponde agli indirizzi da 16384 a 32767. I dati descrittivi degli sprite devono trovarsi nella zona di memoria da 16384 a 32767. I puntatori agli sprite si trovano nei byte di indirizzo 18424-18431 (16384+2040=18424). Se poniamo 14 nel byte di indirizzo 18424, puntatore allo sprite numero 0, questo significa che viene puntato l'indirizzo 16384+14*64=17226. Allo stesso modo variano gli indirizzi dove il VIC II legge le informazioni relative allo sfondo (vedi Paragrafo 2.3.1).

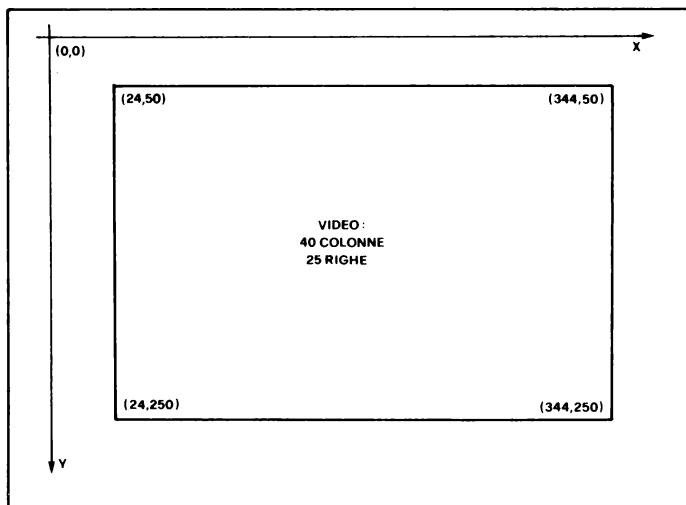



Figura 3.6 Coordinate per il posizionamento degli Sprite

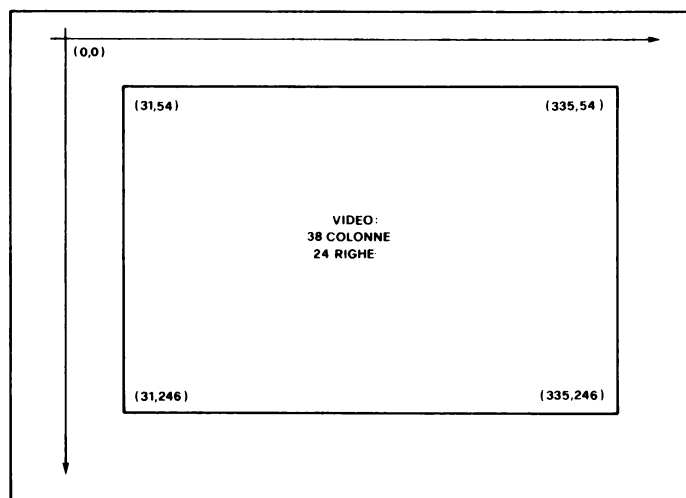



Figura 3.7 Coordinate per il posizionamento degli Sprite

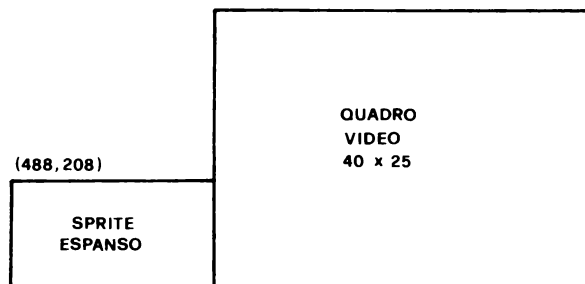



Figura 3.8 Posizione di uno SPRITE ESPANSO

CAPITOLO 4

IL SUONO

4.1 PERCHE' IL SUONO

Il tuo COMMODORE 64 è in grado di produrre suoni; non solo note senza alcun carattere, ma può riprodurre strumenti reali e generare suoni non producibili da alcuno strumento musicale. Così come rende possibile la simulazione dei più diversi rumori. Prima di iniziare ad esplorare tutte le capacità sonore del tuo calcolatore, vogliamo soffermarci un attimo per cercare di rispondere a una domanda: perchè dotare un calcolatore di capacità sonore?
Forse, la prima cosa che viene in mente sono i videogiochi: se a delle buone immagini si associano degli effetti sonori realistici allora il videogioco diventa decisamente più interessante e divertente. Il COMMODORE 64 è in grado di offrire una grafica eccezionale con degli effetti sonori ottimi.
Ma non è questo tutto ciò per cui si può utilizzare il suono.
In programmi comuni esso può servire per il cosiddetto AUDIO FEEDBACK, cioè per segnalare all'utente che il calcolatore ha svolto una determinata azione. Il suono può essere usato per indicare che i dati ricevuti sono errati, oppure che il programma è in attesa di comandi, o ancora, il termine di una lunga elaborazione, per segnalare che i risultati sono pronti, o semplicemente come SUONERIA per un orologio computerizzato. La possibilità di produrre suoni a programma dà all'utente una maggiore libertà di azione; mentre il calcolatore lavora egli può fare altre cose, a un certo punto riceve un avviso e provvede a intervenire.
Se poi ti interessa la musica, con il tuo COMMODORE 64 hai a disposizione 3 generatori indipendenti per creare melodie di tutti i generi; se il diffusore del televisore o del monitor non ti sembra sufficiente per i suoni che vuoi generare, puoi collegare il calcolatore al tuo stereo.
Inoltre suoni ed effetti speciali possono essere ottenuti con delle POKE! Occupiamoci quindi di questo argomento.

4.1.1 Come si genera un suono

L'utilità di dotare un microcalcolatore della capacità di emettere suoni è ormai riconosciuta, ma non tutti lo fanno.
Esistono almeno due modi per permettere a un calcolatore di suonare:

1) Metodo software: il calcolatore è dotato di un altoparlante interno, collegato a un registro o una porta di I/O (cioè un indirizzo al quale si accede per comunicare con dispositivi esterni). Per generare un suono si devono scrivere ripetutamente nelle locazioni adatte valori che creino una sequenza di impulsi, che inviati all'altoparlante ne fanno vibrare la membrana e quindi produrre un suono. Vedi la Figura 4.1.

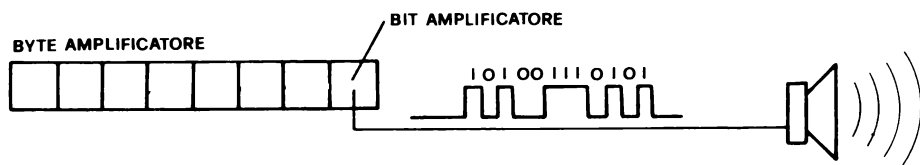



Figura 4.1 Generazione di un suono

2) Metodo hardware: il calcolatore è equipaggiato con una serie di circuiti o con un integrato programmabile in grado di ricevere i dati relativi al suono da produrre e quindi generare un segnale da miscelare al segnale video, e/o un segnale da inviare ad un altoparlante, e/o un segnale prelevabile con apposita presa ed inviabile al proprio HI-FI.
Altri metodi si possono ottenere da una fusione dei due e sono usati in generale nei SINTETIZZATORI e nei CAMPIONATORI.
Per il COMMODORE 64 è stato scelto il secondo metodo. Innanzi tutto esso permette una maggior comodità e velocità nella generazione dei suoni, poichè, una volta selezionati i parametri necessari, è l'integrato che si preoccupa di tutto e il programma può proseguire.
Esso inoltre è generalmente più versatile e più potente. Lo svantaggio è naturalmente nell'aumento del costo del calcolatore, che però la COMMODORE è in grado di evitare, dato che l'integrato sonoro, di codice 6581 e nome SID, è CUSTOM, cioè prodotto dalla ditta per i suoi scopi.
L'accesso alla programmazione del SID è ottenuta facendo in modo che la scrittura in determinati byte ponga i valori scritti nei registri interni del SID, utilizzando la tecnica di MAPPATURA IN MEMORIA del dispositivo, che abbiamo già avuto modo di vedere per il VIC II.

4.1.2 Onde sonore, frequenza, volume e timbro

Immagina di lanciare un sasso in una pozza di acqua; dal punto dove il sasso entra in contatto con il liquido si irradiano delle onde circolari. Viste di profilo queste onde appaiono più o meno come indicato nella Figura 4.2.

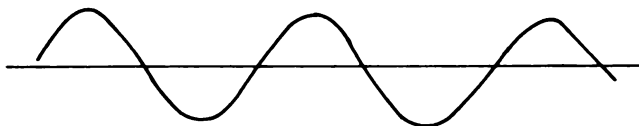



Figura 4.2 Profilo onda sonora

Il suono, come noi lo percepiamo è prodotto da ONDE SONORE che si irradiano da una SORGENTE SONORA in tutte le direzioni.
Mentre le onde dell'acqua mettono il liquido in movimento, le onde sonore sono di COMPRESSIONE e RAREFAZIONE dell'aria, ma si comportano più o meno nello stesso modo delle altre onde; essendo nello spazio la irradiazione risulta su una superficie sferica rispetto alla sorgente.

Se prendi un elastico, lo tendi e lo pizzichi, puoi variare il suono prodotto in 3 modi:

- pizzicando più forte,
- modificando la tensione,
- cambiando l'elastico.

Supponiamo che l'onda sonora prodotta nell'aria dalla vibrazione dell'elastico sia rappresentabile come indicato nella figura 4.3, dove:

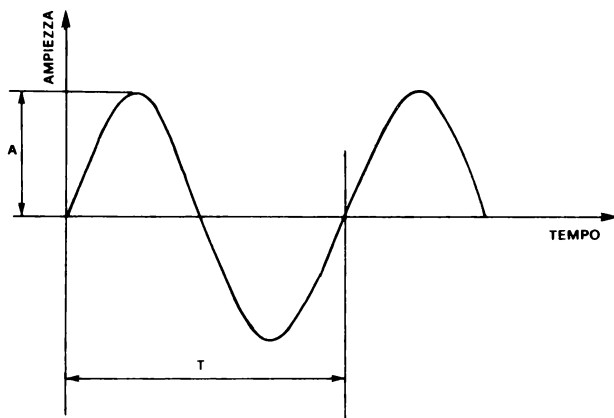



Figura 4.3 Rappresentazione onda sonora

• A, il valore massimo dell'onda, è chiamato AMPIEZZA DELL'ONDA. Se pizzichi l'elastico più forte o più piano, l'ampiezza dell'onda aumenta o diminuisce in conseguenza.

• T è il tempo che intercorre tra due PICCHI delle onde, meglio, il tempo che passa perchè avvenga una vibrazione completa. Per avere il numero di vibrazioni in un secondo basta calcolare 1/T. Questo valore è detto FREQUENZA DELL'ONDA. Se T è espresso in secondi, F=1/T è espresso in Hz. Quando tendi o allenti l'elastico è questa la caratteristica del suono che cambia.

• L'ultima cosa da capire è perchè cambiando elastico e lasciando invariata la tensione e il pizzico, il suono non è uguale al precedente. Il motivo è che i due elastici non sono perfettamente uguali, così come differiscono le corde di una chitarra da quelle di un pianoforte. La caratteristica che ci permette di distinguere due sorgenti sonore, quando i suoni da esse prodotti hanno la stessa frequenza e lo stesso volume, si chiama TIMBRO. Vedremo più avanti come definirlo con IL COMMODORE 64.

4.1.3 Suoni con il Commodore 64

Il centro del sistema sonoro del COMMODORE 64 è un integrato programmabile detto SID (Sound Interface Device, cioè dispositivo di interfaccia per il suono). Come per il VIC II esiste una zona di memoria riservata al SID, ove si possono effettuare le POKE per ottenere i suoni desiderati. Tale zona inizia alla locazione 54272 (D400H) ed è composta da 29 registri, cioè i byte in cui effettuare le POKE. Il SID mette a disposizione 3 diversi canali o VOCI per produrre suoni, che sono totalmente indipendenti tra loro. Per ognuna delle voci sono riservati 7 registri e gli ultimi 8 sono comuni alle 3 voci.
In particolare il registro numero 24 (il 25-esimo), cioè quello all'indirizzo 54296, controlla il volume di tutte le tre voci. Il valore assoluto del volume può variare tra 0 (bassissimo, non elimina completamente il suono) e 15 (il massimo, molto spesso si pone il volume a 15 e poi lo si regola più finemente con il volume del proprio televisore) e la sua variazione si ottiene modificando i bit 0-3 del registro indicato, come puoi vedere nella Figura 4.4, e lasciando inalterati i restanti bit.

```
          BIT     7   6   5   4    3    2    1    0
REG.  N.24      ┌───┬───┬───┬───┬────┬────┬────┬────┐
IND.  54296     │ X │ X │ X │ X │ V3 │ V2 │ V1 │ V0 │
                └───┴───┴───┴───┴────┴────┴────┴────┘
```

Figura 4.4 Registro controllo volume

Se V è il volume e VP il valore precedentemente posto in questo byte:

POKE 54296,VP AND 240 + V

seleziona il volume desiderato.

Torniamo adesso un attimo alla caratteristica del suono chiamata frequenza. L'orecchio umano riesce a percepire suoni la cui frequenza sia compresa tra 20 e 20000 Hz. Il SID è in grado di generare suoni di frequenza compresa tra 0 Hz e 4000 Hz (cioè può generare anche suoni sotto la soglia uditiva umana).
Per indicare al SID la frequenza del suono che si vuole ottenere si deve trasformarla in un numero a 16 bit, compreso tra 0 e 65535. Sono i primi due registri di ogni voce che contengono il byte basso e il byte alto di questo numero a 16 bit. Poichè il valore massimo della frequenza è 4000 Hz e i numeri rappresentabili con 16 bit sono da 0 a 65535, il numero da scrivere in memoria è:

N=Frequenza/(4000/65535)=Frequenza/0.06097
(ovvero N=Frequenza*16.3835)

il che vuol dire che la variazione di N di una unità equivale ad una variazione della frequenza di soli 0.06 Hz.
Ora dobbiamo trovare gli 8 bit più significativi (byte alto) e i meno significativi (byte basso) del numero N:

NHI=INT(N/256) e NLO=N-NHI*256

questi due valori vanno posti negli indirizzi indicati nella Tabella 4.1.

BYTE	VOCE1	VOCE2	VOCE3
NHI	54273	54280	54287
NLO	54272	54279	54286

Tabella 4.1 Indirizzi byte registri frequenza

Nell'Appendice C riportiamo una tabella dei valori utilizzabili per generare quasi 8 ottave di note.
Per calcolarli ci si è basati sul presupposto che la frequenza di ogni nota è circa pari alla radice 12-esima di 2 per la frequenza della nota precedente. Questa è la differenza di frequenza che esiste tra due semitoni, poichè nella scala temperata tra la nota DO1 e la nota DO2 ci sono 12 semitoni e la frequenza di DO2 è doppia di quella di DO1:

```
Freq(DO2)=(2↑12)↑(1/12)*Freq(DO1)=2*Freq(DO1)
```

4.1.4 Forme d'onda e ADSR, il timbro
Il timbro che caratterizza un suono è determinato dalla FORMA DELL'ONDA che si propaga nell'aria, pertanto volume e frequenza non sono sufficienti al SID per produrre un suono; è necessario indicare quale tipo di forma d'onda utilizzare.
Il SID mette a disposizione 4 forme d'onda, come indicato nella Figura 4.5.

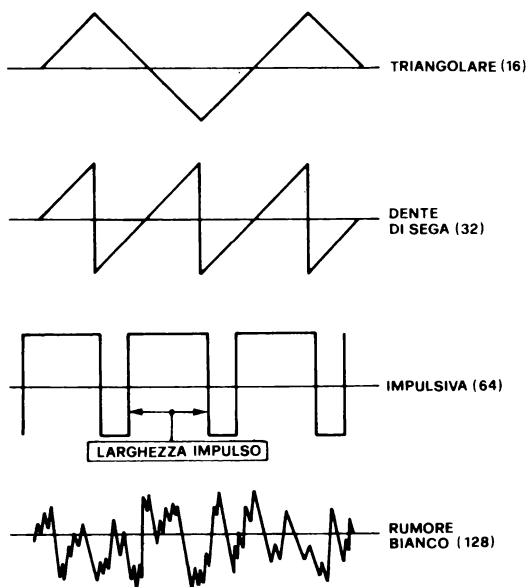



Figura 4.5 Forme d'onda del SID

Per selezionarle devi porre a 1 un determinato bit nel quinto registro della voce che vuoi usare, secondo lo schema che viene riportato nella Figura 4.6.

	VOCE1	VOCE2	VOCE3
FORMA D'ONDA	54276	54283	54290

UTILIZZO BIT							
7	6	5	4	3	2	1	0
●	●	●	●				

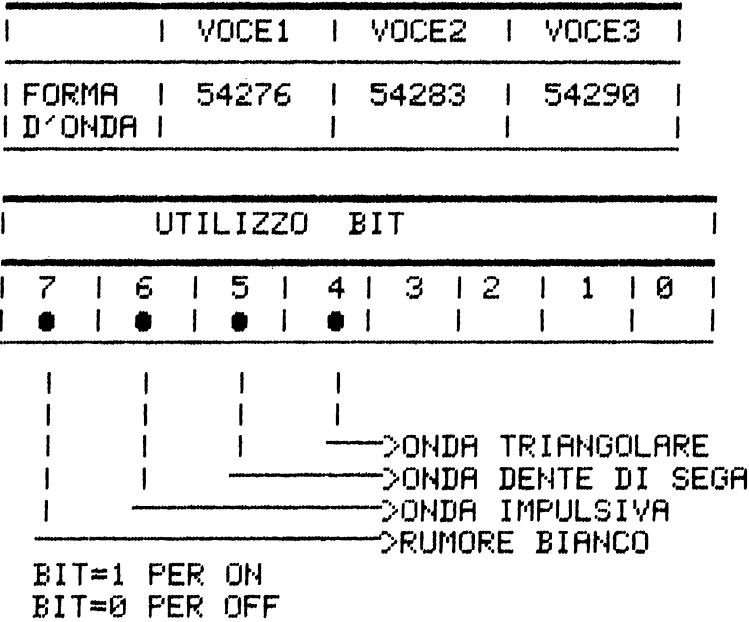



Figura 4.6 Selezione forma d'onda

Se poi selezioni l'onda impulsiva, devi indicare al SID che percentuale dell'onda deve essere positiva; vedi, come riferimento, la Figura 4.7.

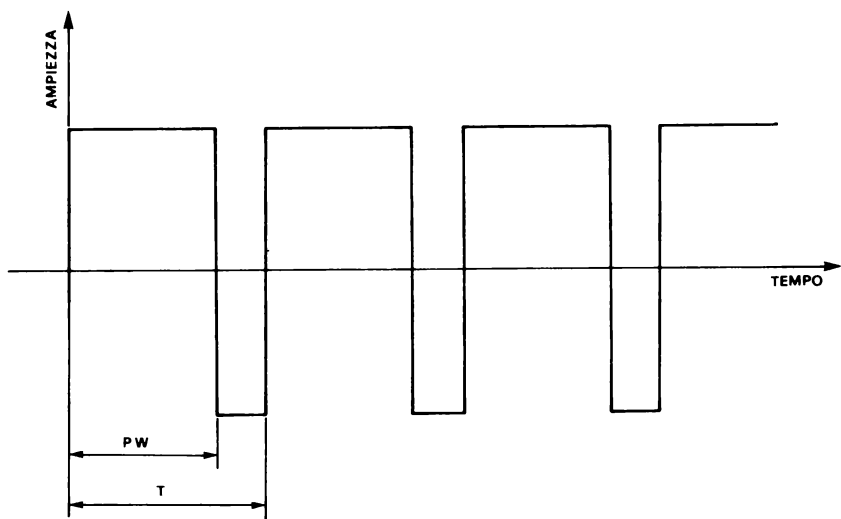



Figura 4.7 Parte positiva impulso

Il rapporto PW=P/T*4095, sotto forma di numero a 12 bit, va trascritto nel terzo e quarto registro per la voce che usi, gli 8 bit meno significativi nel terzo e i 4 più significativi nel quarto. Il valore che si ottiene è:

PW=PWHI * 256 + PWLO

ed in uscita la percentuale positiva dell'impulso è:

(PW/40.95)%

Se PW=0, solo livello basso, o PW=4095 solo livello alto, l'uscita dell'altoparlante sarà nulla. Se PW=2047, cioè PWHI=8 e PWLO=0, l'onda è quadra.
Se R è la percentuale positiva dell'onda impulsiva, allora PW=R*40.95, e le POKE da fare, ad esempio, nel caso della prima voce sono:

POKE 54274, PWLO
POKE 54275, PWHI

Gli indirizzi dei registri da usare in questo caso sono riportati nella Tabella 4.2.

BYTE	VOCE1	VOCE2	VOCE3
PWHI	54275	54282	54289
PWLO	54274	54281	54288

Tabella 4.2 Indirizzi byte dei registri per la parte positiva dell'onda

Vediamo perchè cambiando la forma d'onda cambia il suono.

Quando senti una nota essa è generata da un'onda sinusoidale, la cui frequenza è la frequenza del suono e ne definisce in pratica la tonalità (frequenza fondamentale). Contemporaneamente sono generate altre onde, dette ARMONICHE, la cui caratteristica è avere frequenza multipla per un intero della frequenza fondamentale. Un'onda sonora perciò è la somma della frequenza fondamentale più tutte le armoniche necessarie per produrla.
A ogni armonica è associato l'intero che dà il rapporto con la frequenza fondamentale, cioè l'armonica numero 1. La seconda armonica ha frequenza doppia, la settima armonica ha frequenza sette volte la fondamentale e così via.
Il timbro, cioè la FORMA DELL'ONDA, dipende dalle armoniche che costituiscono il suono e dall'ampiezza di ognuna di esse, quindi tali valori saranno diversi per diverse corde vibranti.
Per le FORME D'ONDA a disposizione valgono queste regole:

- ONDA TRIANGOLARE: solo le armoniche dispari; l'ampiezza di ogni armonica è proporzionale al reciproco del quadrato del numero dell'armonica, cioè, ad esempio, l'armonica numero 5 è ampia 1/25 dell'armonica fondamentale.
- ONDA DENTE DI SEGA: tutte le armoniche; l'ampiezza di ogni armonica è proporzionale al reciproco del numero dell'armonica.
- ONDA QUADRA: solo le armoniche dispari; l'ampiezza di ogni armonica è proporzionale al reciproco del numero dell'armonica (per rapporti R diversi da 50%, cioè per onde rettangolari generiche, la composizione dell'onda impulsiva varia notevolmente).
- RUMORE BIANCO: tutte le armoniche; ampiezza costante per tutte le armoniche.

Per i suoni reali la struttura armonica (cioè le armoniche presenti e la loro ampiezza) è molto più complessa. Per poter ottenere dei suoni più vicini ai reali si può ricorrere alla tecnica di filtrare le frequenze superiori o inferiori ad una certa frequenza data; lo vedremo più avanti.

Un'altra carattetistica che distingue un suono da un altro è il modo di variare di ampiezza nel tempo. Questa caratteristica è detta INVILUPPO del suono (ENVELOPE). Se premi un tasto del pianoforte il suono raggiunge istantaneamente la massima ampiezza ed altrettanto velocemente si smorza, se percuoti un triangolo il suono si smorza più lentamente.
Il SID dispone di un GENERATORE DI INVILUPPO (ENVELOPE GENERATOR) indipendente per ogni voce. Esso permette di descrivere l'andamento del volume del suono dividendolo in 4 fasi:

- A: ATTACCO (ATTACK), è il tempo che il suono impiega per raggiungere il massimo del volume. Con il SID puoi definirlo da 2 msec a 8 sec.
- D: DECADIMENTO (DECAY), è il tempo che il suono impiega, dopo aver raggiunto il picco per calare di intensità fino a zero, oppure fino a un dato livello chiamato di SOSTENIMENTO; da 6 msec a 24 sec.
- S: SOSTENIMENTO (SUSTAIN), è il livello di volume al quale il suono si assesta dopo la fase di decadimento. Puoi mantenere i suoni a questo livello finchè occorre e poi, con una POKE, iniziare la fase di RILASCIO.
- R: RILASCIO (RELEASE), è il tempo che il suono impiega per raggiungere il volume zero. Anche per il rilascio i valori variano da 6 msec a 24 sec.

Molto spesso si fa cominciare una nuova nota prima che la nota precedente sia completamente smorzata. Ciò consente anche, usando adeguatamente le 3 voci, di creare dei veri e propri inseguimenti tra le note.
Nella Figura 4.8 riportiamo il grafico delle 4 fasi e l'inviluppo.

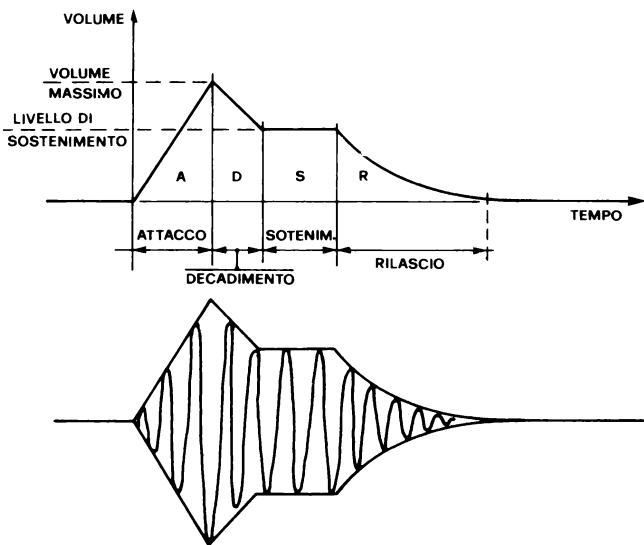



Figura 4.8 Grafico delle 4 fasi e inviluppo

Prima di vedere come indicare al SID il tipo di inviluppo da usare, vogliamo farti notare che puoi ottenere lo stesso effetto utilizzando soltanto il controllo del volume, ma che con questo metodo NON PUOI VARIARE L'ANDAMENTO DEL VOLUME DI OGNI SUONO IN MODO INDIPENDENTE.

Per ognuna delle voci, il sesto e il settimo registro controllano l'ADSR (Attacco, Decadimento, Sostenimento, Rilascio).
Nel sesto registro vanno memorizzati dei valori, compresi tra 0 e 15, per selezionare i tempi di attacco e decadimento secondo la Tabella 4.3.

VALORE	ATTACK TEMPO	DEC/REL TEMPO	SUSTAIN %VOLUME
0	2 MS	6 MS	0.0
1	8 MS	24 MS	6.7
2	16 MS	48 MS	13.3
3	24 MS	72 MS	20.0
4	38 MS	114 MS	26.7
5	56 MS	168 MS	33.3
6	68 MS	204 MS	40.0
7	80 MS	240 MS	46.7
8	100 MS	300 MS	53.3
9	250 MS	750 MS	60.0
10	500 MS	1.5 S	66.7
11	800 MS	2.4 S	73.3
12	1 S	3.0 S	80.0
13	3 S	9.0 S	86.7
14	5 S	15.0 S	93.3
15	8 S	24.0 S	100.0

NOTA: MS PER MILLISECONDI
S PER SECONDI

Tabella 4.3 Valori da porre nel sesto e nel settimo registro

Nel settimo registro vanno memorizzati i valori relativi al sostenimento e al tempo di rilascio; anche questi valori sono riportati nella Tabella 4.3.
Nella Figura 4.9 viene mostrato l'utilizzo dei bit dei due registri.

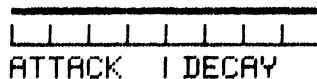
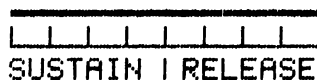



Figura 4.9 Utilizzo sesto e settimo registro

Gli indirizzi dei registri sono mostrati nella Tabella 4.4.

BYTE	VOCE1	VOCE2	VOCE3
A/D	54277	54284	54291
S/R	54278	54285	54292

Tabella 4.4 Indirizzi registri ADSR

La procedura tipica per generare una nota è:

- selezionare ATTACCO, DECADIMENTO, SOSTENIMENTO e RILASCIO per la nota;
- selezionare le frequenze desiderate ed eventualmente la percentuale dell'onda impulsiva;
- selezionare la forma d'onda e mettere a 1 il primo bit dello stesso registro; questo bit, detto GATE BIT, indica che la voce è selezionata per suonare e quando viene riportato a 0 fa iniziare la fase di rilascio. Nella Figura 4.10 sono anche indicati i valori da porre nei registri indicati per le 4 forme d'onda.

	VOCE1	VOCE2	VOCE3
FORMA D'ONDA	54276	54283	54290

UTILIZZO BIT							
7	6	5	4	3	2	1	0
O	O	O	O	-	-	-	●

```
TIPO DI ONDA                              |
                                          |
                                          ->GATE BIT
```

VALORI PER POKE

FORMA D'ONDA	INIZIO ATTACCO	INIZIO RILASCIO
TRIANGOLARE	17	16
DENTE DI SEGA	33	32
IMPULSIVA	65	64
RUMORE BIANCO	129	128

Figura 4.10 Valori per selezionare la forma d'onda

Con un uso appropriato di questo controllo puoi creare effetti sonori più complessi poichè ogni volta che il bit passa da 1 a 0 inizia la fase di rilascio e se passa da 0 a 1 inizia la fase di attacco. I grafici della Figura 4.11 possono servire di chiarimento.

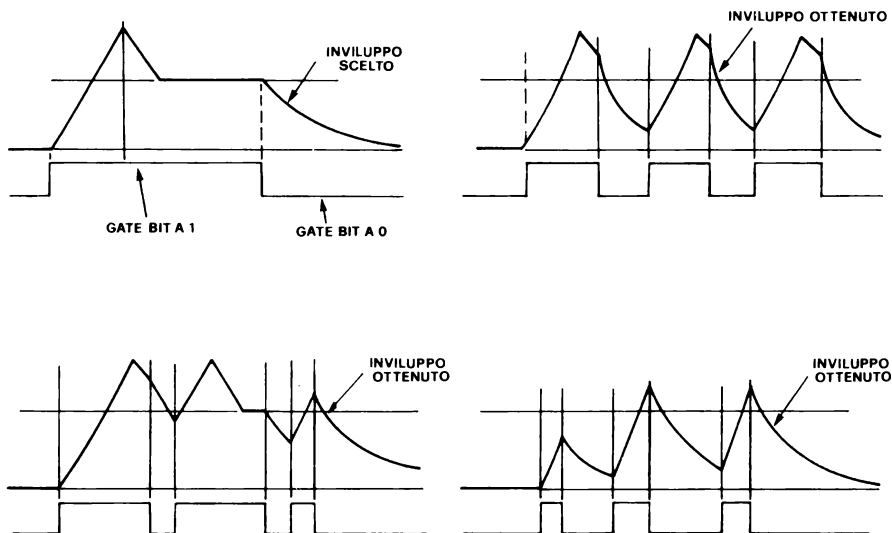



Figura 4.11 Inviluppi

Per sfruttare appieno queste possibilità è però necessario programmare in linguaggio macchina.
Quando vuoi terminare la nota, poni a 0 il GATE BIT.

4.2 PROGRAMMI SONORI

Ora hai tutti gli elementi per comprendere come funziona un programma sonoro. Utilizzando una sola voce è sufficiente, una volta selezionati la forma d'onda e l'inviluppo, fornire la durata e la frequenza di ogni nota, per far suonare il tuo COMMODORE 64.
Ogni coppia di valori XXX,YYYY, che di norma memorizzerai in linee DATA, avrà questo significato:

XXX rappresenta la durata del suono; esso può essere interpretato come parametro finale di un ciclo di conteggio, oppure come numero di 16-esimi o 64-esimi per i quali deve essere tenuta attiva la nota;
YYYY frequenza del suono; risulta più comodo fornirla come 2 numeri che sono i valori da memorizzare nei 2 byte che specificano la frequenza del suono per le varie voci.

Per indicare una pausa si può, ad esempio, usare la frequenza zero o porre la durata e la frequenza come numeri negativi.
Un altro sistema per memorizzare questi dati è di comporre con essi un numero seguendo la regola riportata nella Figura 4.12, esprimendo i parametri in binario e combinandoli insieme.

DURATA	D	OTTAVA	O	NOTA	N
3/2	0	1^	0	PAUSA	0
1	1	2^	1	DO(SI♯)	1
3/4	2	3^	2	DO♯(RE♭)	2
1/2	3	4^	3	RE	3
3/8	4	5^	4	RE♯(MI♭)	4
1/4	5	6^	5	MI	5
3/16	6	7^	6	FA	6
1/8	7	8^	7	FA♯(SOL♭)	7
3/32	8			SOL	8
1/16	9			SOL♯(LA♭)	9
3/64	10			LA	10
1/32	11			LA♯	11
3/128	12			SI	12
1/64	13				

Figura 4.12 Regole per il calcolo dei parametri del suono

In questo caso il numero non supererà mai le 4 cifre decimali (sarà compreso tra 0 e 8191).
Se non devi scendere sotto 1/16 per la durata delle note puoi usare lo stesso sistema:

Valore=((((D*8)+O)*16)+N)
Dove:
D=Durata in 1/16 (di una misura); se negativo indica pausa,
O=Ottava (compresa tra 0 e 7),
N=Nota, tra 0 e 11 o 1 e 12, come preferisci.

Per ricavare i parametri di ogni nota si procede all'inverso; se hai prelevato da linee DATA un VALORE:

Durata=INT(Valore/128)
Ottava=INT((Valore-Durata*128)/16)
Nota= Valore - Durata*128 - Ottava*16

Ti proponiamo, come esempio, il programma SUONO4.

```
1 REM SUONO4
10 REM FARE MUSICA
100 S=54272:FORI=STOS+24:POKEI,0:NEXT
110 DIMF(11)
120 POKES+3,8:REMPOKES+22,128:POKES+23,244
130 F(11)=67280:FORI=10TO0STEP-1
140 F(I)=INT(F(I+1)/2↑(1/12)+.5)
150 NEXT
160 O=5
170 POKES+5,41:POKES+6,0
180 POKES+24,31
190 READN$,T:IFN$="*"THEN430
200 PRINTN$;T;
210 IFN$="O"THENO=T:GOTO190
220 IFN$="O+"THENO=O+1AND7:GOTO190
230 IFN$="O-"THENO=O-1AND7:GOTO190
240 IFN$="R"THENRESTORE:GOTO190
250 IFN$="DO"THENN=0
260 IFN$="DO#"THENN=1
270 IFN$="RE"THENN=2
280 IFN$="RE#"THENN=3
290 IFN$="MI"THENN=4
300 IFN$="FA"THENN=5
310 IFN$="FA#"THENN=6
320 IFN$="SOL"THENN=7
```

```
330 IFN$="SOL#"THENN=8
340 IFN$="LA"THENN=9
350 IFN$="SI"THENN=11
360 F=F(N)/2↑(7-O)
365 IFF>65535THENF=0:REM SI-7
370 POKES,F-INT(F/256)*256
380 POKES+1,F/256
390 POKES+4,64:POKES+4,65
400 FORR=1TO2↑T*5:NEXT
420 GOTO190
430 FORL=STOS+24:POKEL,0:NEXT
440 DATAO-,0,DO,3,RE,2,MI,3,SOL,3,SOL,3,LA
445 DATA2,SOL,3,MI,4,DO,2,RE,3,MI,3,MI,3,RE,2
450 DATADO,2,RE,6,DO,3,RE,2,MI,3,SOL,3,SOL
455 DATA3,LA,2,SOL,3,MI,4,DO,2,RE,3,MI,3,MI,3
460 DATARE,2,RE,2,DO,6,FA,5,FA,5,LA,3,LA
465 DATA5,LA,2,SOL,3,SOL,3,MI,3,DO,3,RE,6
470 DATADO,3,RE,2,MI,3,SOL,3,SOL,3,LA,2,SOL
475 DATA3,MI,4,DO,2,RE,3,MI,3,MI,3,RE,2,RE,2
480 DATADO,6,R,4
```

COMMENTO A SUONO4.
Linea 100: inizializza la variabile S con l'indirizzo del SID.
Linea 110: il vettore F viene usato per contenere i valori da porre nel SID per generare i semitoni più alti.
Linea 120: pone la larghezza dell'impulso dell'onda quadra a metà ciclo e il valore del filtro a 3520 Hz.
Linea 130: F(11) contiene il numero da porre nel SID per generare il SI dell'ottava più alta (ottava 7); tale valore supera 65535, perchè nella versione europea la nota più alta che il SID può generare è il LA #. La linea 365 porrà a 0 la frequenza, nel caso in cui si voglia suonare un SI nell'ottava 7.
Linee 140-150: il valore della frequenza di un semitono si ricava da quello del semitono superiore, diviso per la radice 12-esima di 2.
Linea 160: pone come ottava di default l'ottava 5.
Linea 170: pone ATTACK=2, DECAY=9, SUSTAIN=0 e RELEASE=0.
Linea 180: pone a 15 il volume e aziona il filtro passa basso.
Linea 190: N$ e T rappresentano i valori della nota e la sua durata; N$, oltre ai nomi delle 7 note può assumere i significati:

O+, passaggio all'ottava superiore,
O-, passaggio all'ottava inferiore,

O, passaggio all'ottava specificata,
R, ripetizione dall'inizio,
*, termine dell'esecuzione.

Linea 200: evidenzia sul video il valore della nota corrente.
Linee 210-350: assegna a N il valore del semitono che corrisponde alla nota N$, o esegue il comando che N$ rappresenta.
Linea 360: calcola la frequenza in base alla nota e all'ottava; ogni nota ha infatti la metà della frequenza della stessa nota nell'ottava superiore.
Linea 365: se la frequenza della nota è troppo alta per il SID, allora viene annullata. Tale caso si verifica solo con il SI dell'ottava 7.
Linee 370-380: pone nei registri della frequenza i valori calcolati.
Linea 390: suona la nota.
Linea 400: attende per un tempo variabile, che dipende da T. T rappresenta la durata della nota, secondo questi criteri:

T=0, un 32-esimo
T=1, un 16-esimo
T=2, un ottavo
T=3, un quarto
T=4, un mezzo
T=5, un intero.

Linea 420: va a suonare la prossima nota.
Linea 430: termina l'esecuzione azzerando i registri del SID.
Linee 430-470: linee DATA che contengono un semplice motivo. Il formato dei dati è il seguente: nome della nota, tempo. Le note possono avere il diesis; se si introduce al posto della nota un comando, che non ha bisogno del parametro tempo, bisogna ugualmente porre un numero come tempo, per consentire la lettura dei dati in coppia.

Utilizzando più voci insieme è possibile ottenere suoni più pieni, accordi e l'impressione di orchestra poichè ogni voce può simulare uno strumento diverso. L'unico problema vero è quello della sincronizzazione delle 3 voci.
Anche in questo caso le soluzioni sono molte e diverse. La più pratica sembra però quella descritta anche nella Programmer's Reference Guide, cioè, per ognuna delle 3 voci, preparare una descrizione di ciò che avviene ogni 16-esimo di battuta (o ottavo o 64-esimo, dipende dall'uso che ne vuoi fare), quindi, stabilita la durata del 16-esimo, si legge la sequenza dei dati e si trascrivono nei registri delle 3 voci.

Figura 4.13 Sequenza di note

Ad esempio, se il pezzo da suonare con una voce è quello riportato nella Figura 4.13, le note sono: FA2, RE2, SI2, DO2 (il numero indica l'ottava), le durate 1/4, 1/4, 1/8, 1/8 (cioè 4/16, 4/16, 2/16, 2/16) e si ottiene uno schema in 16-esimi, come indicato nella Figura 4.14.

SEDICESIMI	NOTA	OTTAVA	ULT. SED.
1	FA	4	
2	FA	4	
3	FA	4	
4	FA	4	●
5	RE	4	
6	RE	4	
7	RE	4	
8	RE	4	●
9	SI	3	
10	SI	3	●
11	DO	4	
12	DO	4	●

Figura 4.14 Schema in sedicesimi di una sequenza di note

Esso può essere codificato con 12 triplette di numeri che indichino i valori da porre nei byte che definiscono la frequenza (2 byte) e il valore da porre nel byte che determina la forma d'onda. Dopo l'ultimo 16-esimo di ogni nota (nello schema sono indicati con un puntino) la nota verrà fatta terminare scrivendo nel registro che regola la forma d'onda un valore che ponga a 0 il GATE BIT della voce specificata (con un AND 254 del valore da scrivere nel registro apposito).

La fase più noiosa di tutto ciò sta nella codifica delle note che, già semplificata con i due metodi visti all'inizio del paragrafo, può essere ulteriormente agevolata costruendo un programma che accetti i dati per le varie voci dall'utente, ne consenta modifiche, cancellazioni, memorizzazione e ricaricamento, e li trasformi quindi:

- o nei dati organizzati in 3 vettori (byte alto frequenza, byte basso frequenza, forma d'onda), che è la più comoda se si utilizzano le 3 voci insieme;
- in istruzioni più semplici ed eseguite più rapidamente da una parte del programma che si occupa solo di suonare.

Entrambi i sistemi vengono notevolmente migliorati dall'uso del linguaggio macchina.

4.3 L'USO DEI FILTRI

Abbiamo visto che nella definizione di un suono entrano in gioco molti parametri. In particolare scegliendo una forma d'onda piuttosto che un'altra (e con l'onda rettangolare ne puoi ottenere molte tutte diverse), cambia la struttura armonica dell'onda sonora generata. L'uso della tecnica chiamata FILTERING (in italiano diventa filtraggio) permette di eliminare SELETTIVAMENTE le frequenze intorno, al di sopra o al di sotto di una frequenza base, detta frequenza di taglio (o rottura). Vediamo di spiegarci meglio: un FILTRO è un circuito che, ricevendo in ingresso un segnale è in grado di trasmetterlo (riprodurlo) modificato, eliminando un certo gruppo di frequenze che costituiscono il segnale oppure togliendo la parte di segnale di frequenza superiore o inferiore ad un certo limite, o altre cose del genere. Le caratteristiche che distinguono i diversi tipi di filtro sono:

- la frequenza di taglio,
- il tipo di filtro,
- il grado di smorzamento delle altre frequenze.

TIPO DI FILTRO indica quali frequenze, rispetto a quella di taglio devono essere eliminate, questa classificazione dà i filtri passa basso, passa alto, passa banda, taglia banda (o a tacca, notch in inglese) come indicato nella Figura 4.15.

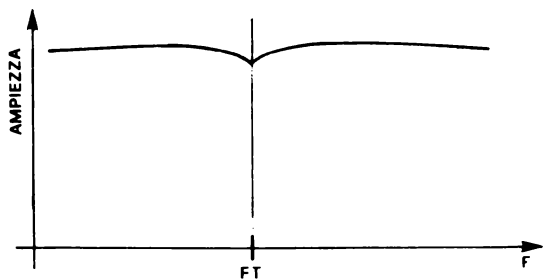
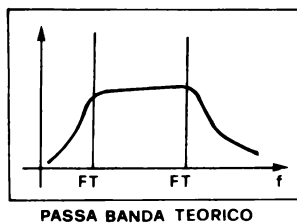
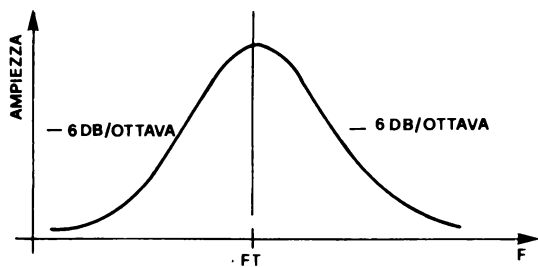
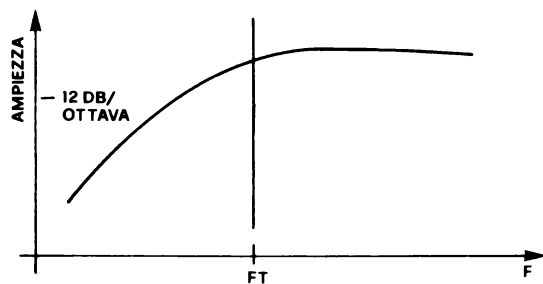
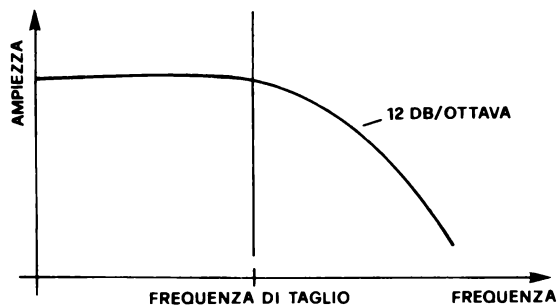



Figura 4.15 Tipi di filtri

Il COMMODORE 64 è equipaggiato con i filtri: passa basso, passa alto e passa banda, che, essendo adittivi, possono essere combinati per ottenere il filtro a tacca. Nella Figura 4.15 sono indicati i grafici dei 4 tipi di filtri.
Nel FILTRO PASSA BASSO vengono attenuate le frequenze al di sopra di quella di taglio. Nel FILTRO PASSA ALTO vengono attenuate le frequenze al di sotto di quella di taglio. Nel FILTRO PASSA BANDA vengono attenuate sia le frequenze al di sopra che quelle al di sotto della frequenza di taglio. Nel FILTRO A TACCA vengono attenuate la frequenza di taglio e quelle nel suo intorno; esso si ottiene attivando contemporaneamente i filtri passa basso e passa alto.

La PENDENZA delle curve dei grafici indica il grado di attenuazione delle frequenze da escludere. Per il SID l'attenuazione è di 12 db/ottava per i filtri passa alto e passa basso e di 6 db/ottava per il filtro passa banda. Osserva in proposito le note della Figura 4.15.

La FREQUENZA DI TAGLIO, che può variare tra circa 30 Hz e 12 KHz si specifica al SID come un numero a 11 bit (0-2047), quindi la variazione di unità corrisponde a una differenza della frequenza di taglio di circa 6 Hz, sufficientemente piccola per la maggior parte delle applicazioni.
Il numero a 11 bit va spezzato in modo da porre i 3 bit meno significativi nel byte 54293 (registro numero 21, il 22-esimo) e gli 8 bit più significativi nel byte 54294 (registro numero 22, il 23-esimo), come indicato nella Figura 4.16.

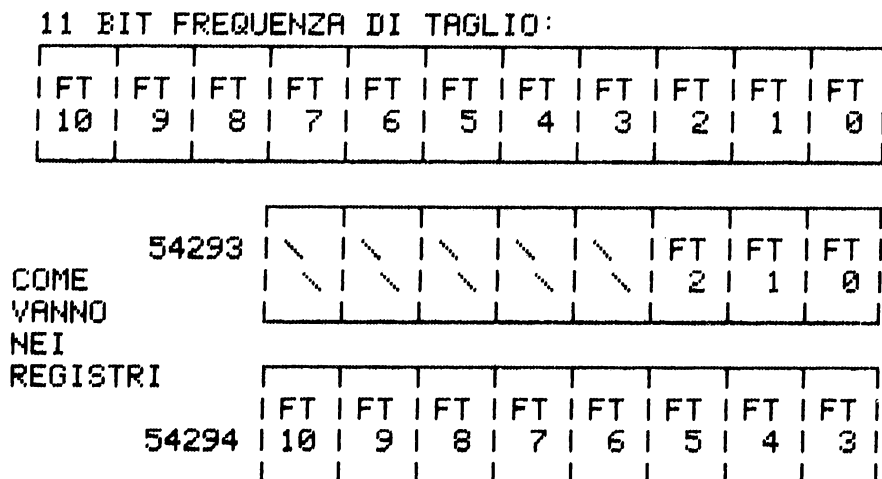



Figura 4.16 Frequenza di taglio

In pratica se FT è la frequenza di taglio si devono eseguire le due POKE indicate:

POKE 54293, FT AND 7
POKE 54294, INT(FT/8)

Non cercare di rileggere i valori posti nei byte, dato che non li otteresti uguali a quelli scritti poichè i registri del SID sono a sola scrittura.

Come per il volume, anche il filtro è comune a tutte 3 le voci. E' perciò impossibile selezionare un filtro diverso per ogni voce, mentre è possibile scegliere quali voci devono essere filtrate e quali no.
Per selezionare un determinato tipo di filtro si deve porre a 1 il bit corrispondente nel registro numero 24 (il 25-esimo) del SID, all'indirizzo 54296 (lo stesso del volume). Osserva la Figura 4.17.

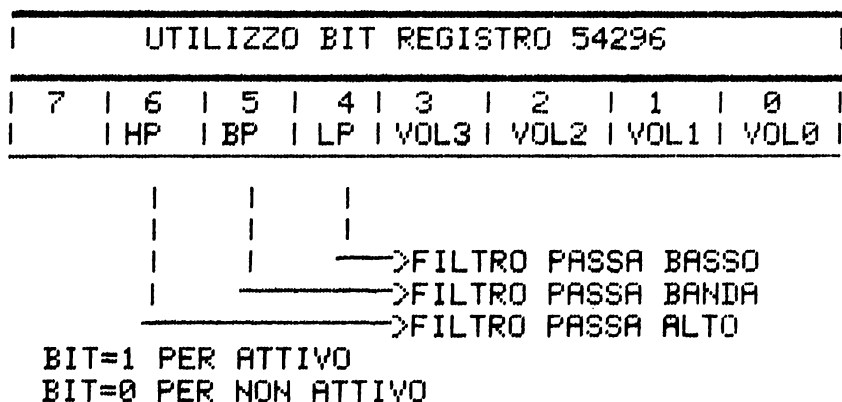



Figura 4.17 Utilizzo registro di indirizzo 54296

Poichè anche questo registro non può essere riletto, per selezionare i vari filtri, se V è il volume (da 0 a 15), si devono usare le istruzioni:

POKE 54296, V OR 16	per attivare il filtro passa basso
POKE 54296, V OR 32	per attivare il filtro passa banda
POKE 54296, V OR 64	per attivare il filtro passa alto
POKE 54296, V OR 80	per attivare il filtro a tacca (80=16+64)

Per indicare, invece, quali voci vanno filtrate, si utilizzano i bit meno significativi del registro numero 23, cioè del byte di indirizzo 54295, come indicato nella Figura 4.18.

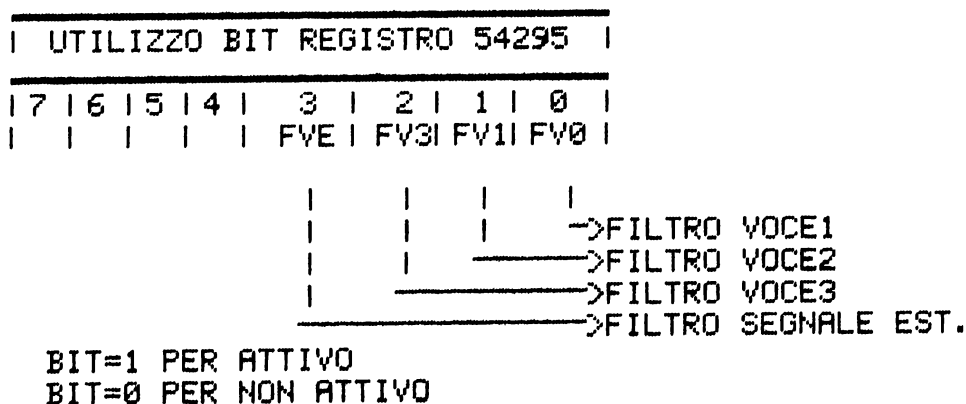



Figura 4.18 Utilizzo registro di indirizzo 54295

Il quarto bit di questo stesso registro indica se anche il segnale esterno, che è connesso al SID, deve essere filtrato. Infatti il SID è in grado di ricevere un segnale dall'esterno e fonderlo con quelli delle 3 voci, permettendo effetti polifonici collegando tra loro le uscite di più SID oppure il segnale da un HI-FI.
A noi interessa relativamente; normalmente possiamo trascurare i 4 bit più significativi del registro numero 23 (il 24-esimo) e usare l'istruzione:

POKE 54295, FV

dove per FV vale quanto esposto nella Tabella 4.5. Se esiste un segnale esterno, per ottenere il filtraggio è sufficiente aggiungere 8 al valore di FV.

VALORE FV	VOCI FILTRATE 1	2	3
0	NO	NO	NO
1	SI	NO	NO
2	NO	SI	NO
3	SI	SI	NO
4	NO	NO	SI
5	SI	NO	SI
6	NO	SI	SI
7	SI	SI	SI

Tabella 4.5 Valori per registro 54295

Il SID ci permette però di aggiungere un altro effetto, legato alla frequenza di taglio indicata per i filtri.

4.3.1 La risonanza

La risonanza è un'esaltazione, un aumento di ampiezza, delle frequenze prossime alla frequenza di taglio. Ad esempio, per un filtro passa alto, l'andamento dell'ampiezza delle frequenze sarebbe quella indicata nella Figura 4.19.

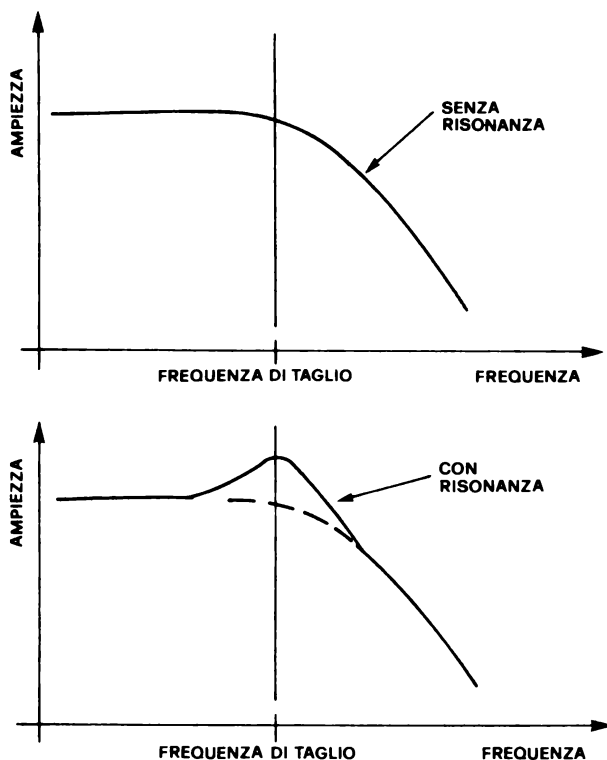



Figura 4.19 Andamento ampiezza frequenza per filtro passa alto

Questo effetto permette di ottenere suoni più decisi, privilegiando la frequenza di taglio sulle altre frequenze. Con il SID puoi graduare l'effetto di risonanza. Per ottenerlo, infatti, devi porre un valore nei 4 bit più significativi del registro numero 23 (il 24-esimo), appena visto per l'uso dei filtri.
Tale valore sarà compreso tra 0 (niente risonanza) e 15 (massima risonanza) e andrà scritto nel byte 54295 con:

POKE 54295, R*8+FV
dove:

R=0...15 è il valore per la risonanza
FV=0...15 è il numero che indica le voci da filtrare.

Osserva la Figura 4.20.

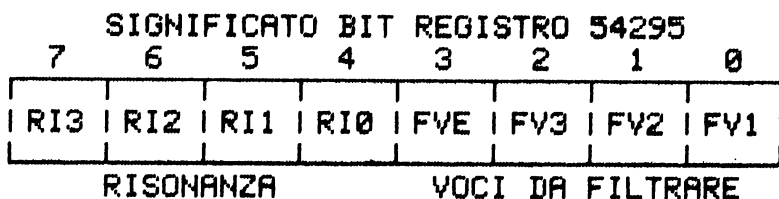



Figura 4.20 Utilizzo registro di indirizzo 54295

4.4 ANCORA DI PIU'

Nell'esaminare le capacità sonore del SID abbiamo visto come usare i diversi registri, ma se ci fai caso vedi che sono rimasti dei buchi. La mappa, riportata nella Figura 4.21, ti indica quali registri e quali bit abbiamo già esaminato.

REG. SID	IND. DEC.	SIGNIFICATO BIT 7	6	5	4	3	2	1	0
0	54272	F7	F6	F5	F4	F3	F2	F1	F0
1	54273	F15	F14	F13	F12	F11	F10	F9	F8
2	54274	PW7	PW6	PW5	PW4	PW3	PW2	PW1	PW0
3	54275	--	--	--	--	PW11	PW10	PW9	PW8
4	54276	RB	OI	ODS	OT	●	●	●	GATE
5	54277	A3	A2	A1	A0	D3	D2	D1	D0
6	54278	S3	S2	S1	S0	R3	R2	R1	R0
7	54279	F7	F6	F5	F4	F3	F2	F1	F0
8	54280	F15	F14	F13	F12	F11	F10	F9	F8
9	54281	PW7	PW6	PW5	PW4	PW3	PW2	PW1	PW0
10	54282	--	--	--	--	PW11	PW10	PW9	PW8
11	54283	RB	OI	ODS	OT	●	●	●	GATE
12	54284	A3	A2	A1	A0	D3	D2	D1	D0
13	54285	S3	S2	S1	S0	R3	R2	R1	R0
14	54286	F7	F6	F5	F4	F3	F2	F1	F0

15	54287	F15	F14	F13	F12	F11	F10	F9	F8
16	54288	PW7	PW6	PW5	PW4	PW3	PW2	PW1	PW0
17	54289	--	--	--	--	PW11	PW10	PW9	PW8
18	54290	RB	OI	ODS	OT	●	●	●	GATE
19	54291	A3	A2	A1	A0	D3	D2	D1	D0
20	54292	S3	S2	S1	S0	R3	R2	R1	R0
21	54293	--	--	--	--	--	FT2	FT1	FT0
22	54294	FT10	FT9	FT8	FT7	FT6	FT5	FT4	FT3
23	54295	RI3	RI2	RI1	RI0	FVE	FV3	FV2	FV1
24	54296	●	HP	BP	LP	VOL3	VOL2	VOL1	VOL0
25	54297	O	O	O	O	O	O	O	O
26	54298	O	O	O	O	O	O	O	O
27	54299	●	●	●	●	●	●	●	●
28	54300	●	●	●	●	●	●	●	●

NOTE:

F PER FREQUENZA SUONO
PW PER PARTE POSITIVA IMPULSO
RB PER RUMORE BIANCO
OI PER ONDA IMPULSIVA
ODS PER ONDA DENTE DI SEGA
OT PER ONDA TRIANGOLARE
A PER ATTACK
D PER DECAY
S PER SUSTAIN
R PER RELEASE
FT PER FREQUENZA DI TAGLIO
RI PER RISONANZA
FV PER FILTRO
FVE PER FILTRO SEGNALE ESTERNO
HP PER FILTRO PASSA ALTO
BP PER FILTRO PASSA BANDA
LP PER FILTRO PASSA BASSO
VOL PER VOLUME

Figura 4.21 Mappa dei registri del SID

I tondini pieni indicano bit e byte di cui non abbiamo visto l'utilizzo e di cui ci occupiamo ora.

4.4.1 Modulazione ad anello e sincronizzazione

I registri numero 4, 11 e 18 (rispettivamente quinto, 12-esimo e 19-esimo, di indirizzi 54276, 54283 e 54290) oltre a permettere di selezionare la forma d'onda e attivare il generatore di inviluppo per ogni voce (vedi Paragrafo 4.1.2), contengono altri 3 bit, come indicato nella Figura 4.22.

Figura 4.22 Significato bit registri numero 4, 11 e 18

RING MODULATION. Quando il bit 2 dei registri indicati è posto a 1 e contemporaneamente è selezionata l'onda triangolare, l'uscita della voce a cui è associato il registro è sostituita dalla modulazione ad anello (RING MOD.):
registro 4, voce 1 modulata con voce 3
registro 11, voce 2 modulata con voce 1
registro 18, voce 3 modulata con voce 2
le frequenze devono essere diverse da zero.
La modulazione ad anello permette di variare ulteriormente la struttura armonica dei suoni generati, aggiungendo frequenze non armoniche e permettendo ulteriori mutamenti variando la frequenza del generatore associato. Ma è più facile capirlo provando che leggendo le spiegazioni. Si rivela molto utile per ottenere effetti tipo gong o altri suoni metallici.

SYNC. Quando si pone a 1 il bit 1 dei 3 registri indicati si ottiene la sincronizzazione tra la frequenza fondamentale della voce del registro e la frequenza fondamentale di un'altra voce (valgono le stesse relazioni della modulazione ad anello), che deve essere diversa da zero e possibilmente minore della prima. Anche in questo caso la variazione (meglio se in tempo reale) dei valori della frequenza della voce associata permette di ottenere strutture armoniche più complesse di quelle di partenza.
La sincronizzazione, in pratica, è lo AND logico tra le due frequenze di base, per cui la risultante è zero quando una delle due è zero.

TEST. Porre a 1 questo bit BLOCCA e RESETTA (pone a zero i registri) la voce che è controllata da questo registro. Il che vuol dire che cessano tutte le funzioni di tale generatore sonoro, fino a quando lo stesso bit non viene riportato a zero. Può essere comodo usato in un programma in quanto è un metodo più veloce che fare 24 POKE nell'area dei registri. Utile anche, nella programmazione in linguaggio macchina, per sincronizzare il SID (una voce del SID) con un evento esterno.

4.4.2 Cambiamenti dinamici del suono

Nel registro numero 24 (il 25-esimo, volume e filtri) c'è un bit che non abbiamo mai usato, quello di posizione 7. Osserva la Figura 4.23.

```
       SIGNIFICATO BIT REGISTRO 54296
  7     6     5     4     3     2     1     0
 ___________________________________________________
I  3  I  HP I  BP I  LP I VOL I VOL I VOL I VOL I
I OFF I     I     I     I  3  I  2  I  1  I  0  I
I_____I_____I_____I_____I_____I_____I_____I_____I
```

Figura 4.23 Bit di posizione 7 del registro 54296

3 OFF: Porre 1 in questo bit produce l'effetto di eliminare qualunque uscita dal generatore numero 3, impedire cioè che la voce 3 possa emettere suoni. Quale la sua funzione pratica?

Abbiamo visto che il SID è mappato in memoria e che si accede ai suoi registri con delle POKE. Dei suoi 29 registri 25 sono A SOLA SCRITTURA (WRITE ONLY), cioè tentativi di lettura, per motivi di mancanza di connessioni, non restituiscono i valori realmente contenuti nei registri, e, a questo punto, li abbiamo già esaminati tutti. Restano 4 registri, che SONO A SOLA LETTURA. I primi 2 permettono di leggere (digitalizzato) lo stato di due potenziometri (numero 25, il 26-esimo di indirizzo 54297, e numero 26, il 27-esimo di indirizzo 54298), tipicamente due paddle, ma non solo quello.

Il registro numero 27 (il 28-esimo di indirizzo 54299), indicato nelle mappe come OSC3/RANDOM, riflette in ogni istante l'andamento dell'uscita audio della voce 3, a seconda della forma d'onda selezionata per tale voce:

• se hai selezionato l'onda triangolare, il contenuto del registro numero 27 varia da 0 a 255, poi scende a 0 ciclicamente, con una velocità che dipende SOLO dalla frequenza scelta;

• se hai selezionato l'onda a dente di sega, il valore varia da 0 a 255 e torna immediatamente a 0 per riprendere a salire;

• se hai selezionato l'onda impulsiva ottieni una sequenza di 0 e 255 corrispondenti con le fasi negative e positive dell'onda;

• la selezione del rumore bianco fa comparire nel registro numeri a caso tra o e 255.

In tutti i casi l'inviluppo scelto per la voce 3 non influenzerà i risultati delle letture, inoltre la frequenza selezionata per questa voce deve essere diversa da zero.
Il registro numero 28 (il 29-esimo di indirizzo 54300), invece, riflette proprio i mutamenti nell'ampiezza dell'uscita audio della voce 3, cioè dell'andamento ADSR.
Entrambi i registri (con i loro contenuti) possono essere utilizzati per creare effetti speciali come il TREMOLO, lo WAH-WAH, le sirene, e altri ancora, fornendo un ulteriore modo per modificare i suoni prodotti in base dalle voci del SID.
Poichè spesso l'uscita audio della voce 3 sarà indesiderata, in quanto sarà stata programmata (la voce 3) per leggere i valori dai due registri appena visti è stato reso possibile INIBIRNE l'uscita, con una semplice POKE nel registro numero 24, che ponga a 1 il bit di posizione 6.

4.5 PER CONCLUDERE

Abbiamo visto praticamente tutte le possibilità del SID, che, usate con un pò di fantasia, permettono la generazione di una varietà notevolissima di suoni. Per finire ti proponiamo ancora qualche esempio in cui ritroverai utilizzate le informazioni viste nei paragrafi precedenti.

Il programma SUONO1 è una dimostrazione delle capacità AVANZATE del SID: filtri, modulazione ad anello, sincronizzazione e variazione dinamica del suono.

```
1 REM SUONO1
10 POKE53280,0:POKE53281,0
15 PRINTCHR$(154)CHR$(147)
20 V1=54272:V3=54286
30 PRINT"ONDA TRIANGOLARE"
40 POKEV1,0
50 POKEV1+1,22
60 POKEV1+5,0
```

```
70 POKEV1+6,240
80 POKEV1+4,17
90 POKEV1+24,15
100 FORI=1TO2000:NEXT
130 PRINT:PRINT"SINCRONIZZATA"
150 POKEV3,0
160 POKEV3+1,10
170 POKEV3+4,16
200 POKEV1+4,19
210 FORI=1TO2000:NEXT
220 PRINT:PRINT"E MODULATA"
230 POKEV1+4,21
240 FORI=1TO2000:NEXT
250 PRINT:PRINT"TREMOLO"
260 POKEV1+4,17
280 POKEV3,60:POKEV3+1,0
300 FORI=1TO300:POKEV1+24,PEEK(54299)/16:NEXT
400 PRINT:PRINT"FILTRO PASSA BASSO"
410 POKEV1+24,31:POKEV1+23,1:POKEV1+22,19
420 FORI=1TO2000:NEXT
430 PRINT:PRINT"FILTRO PASSA BANDA"
440 POKEV1+24,47
450 FORI=1TO2000:NEXT
460 PRINT:PRINT"FILTRO PASSA ALTO"
470 POKEV1+24,79
480 FORI=1TO2000:NEXT
481 PRINT:PRINT"FILTRO A TACCA"
482 POKEV1+24,95
483 FORI=1TO2000:NEXT
490 PRINT:PRINT"VIBRATO"
500 POKEV3,22:POKEV1+24,47
510 FORI=1TO500:POKEV1+22,PEEK(54299)/2:NEXT
520 FORI=0TO24:POKEV1+I,0:NEXT
```

COMMENTO A SUONO1.
Linea 10: inizializza il video.
Linee 20-100: produce un'onda triangolare.
Linee 130-210: sincronizza la voce 1 con la voce 3, ponendo nel registro di controllo della voce 1 (54276 (D404H)) il bit di posizione 1 a 1.
Linee 220-240: modula la voce 1 con la voce 3, ponendo nel registro di controllo della voce 1 (54276 (D404H)) il bit di posizione 2 a 1.

Linee 250-300: ottiene un effetto di tremolo variando dinamicamente il suono. Pone cioè nel registro del volume (54296 (D418H)) il contenuto del registro 54299 (D41BH) che contiene il valore digitale dell'uscita della voce 3. Tale valore (che varia da 0 a 255) è stato diviso per 16 poichè il volume può variare solo tra 0 e 15.
Linee 400-420: aziona il filtro passa basso, ponendo:

a 1 il bit di posizione 4 del registro 54296 (D418H) per selezionare il filtro passa basso;

a 1 il bit di posizione 0 del registro 54295 (D417H) per filtrare la voce 1;

19 nel registro 54294 (D416H) per ottenere come frequenza di taglio la stessa frequenza a cui oscilla la voce 1.

Linee 430-450: aziona il filtro passa banda ponendo a 1 il bit di posizione 5 del registro 54296 (D418H) per selezionare tale filtro.
Linee 460-480: aziona il filtro passa alto ponendo a 1 il bit di posizione 6 del registro 54296 (D418H) per selezionare tale filtro.
Linee 481-483: aziona il filtro a tacca ponendo a 1 i bit di posizione 4 e 6 del registro 54296 (D418H) per selezionare i filtri passa basso e passa alto.
Linee 490-520: ottiene un effetto vibrato variando dinamicamente il suono come per l'effetto tremolo, ma agendo sulla frequenza di taglio del filtro passa banda, anzichè sul volume.
Il programma SUONO2 dimostra come si possa usare completamente la voce 3 (sia oscillatore che generatore di inviluppo) per ottenere una variazione dinamica del suono. La frequenza di uscita del generatore 1 è proporzionale all'uscita che avrebbe la voce 3 se non fosse disabilitata dal bit di posizione 7 del registro 54296 (D418H).

```
1 REM SUONO2
5 REM CAMBIAMENTO DINAMICO CON VOCE 3
10 V1=54272:V3=54286
20 POKEV3,20
30 POKEV3+1,0
40 POKEV3+5,240
50 POKEV3+6,253
60 POKEV1+24,15+128
70 POKEV1,0
80 POKEV1+5,0:POKEV1+6,240
90 POKEV1+4,33
100 POKEV3+4,17
110 FORI=0TO400
115 POKEV1+1,PEEK(V1+27)/255*PEEK(V1+28):NEXT
120 POKEV3+4,16
130 FORI=0TO500
135 POKEV1+1,PEEK(V1+27)/255*PEEK(V1+28):NEXT
```

COMMENTO A SUONO2.
Linee 20-50: seleziona per la voce 3 una bassa frequenza di oscillazione, attacco=15, decadimento=0, sostenimento=15, rilascio=13.
Linea 60: pone il volume a 15 e disabilita la voce 3.
Linee 70-90: seleziona per la voce 1: attacco=0, decadimento=0, sostenimento=15, rilascio=0. Pone a 0 il byte meno significativo della frequenza, seleziona la forma d'onda a dente di sega e fa partire l'oscillatore.
Linea 100: seleziona per la voce 3 la forma d'onda triangolare e fa partire l'oscillatore.
Linee 110-115: pone nel byte più significativo della frequenza della voce 1 il valore che avrebbe l'uscita della voce 3 se non fosse disabilitata.
Linee 130-135: fa partire la fase di rilascio della voce 3 e continua con il calcolo della frequenza della voce 1 per 500 cicli.

Il programma SUONO3 dimostra come si possa ottenere una variazione dinamica del suono senza appoggiarsi alla voce 3, ma, per esempio, a ciò che accade sul video. In questo caso viene visualizzata una pallina che rimbalza; la frequenza del suono emesso dal SID è proporzionale all'altezza della pallina e il volume alla posizione orizzontale della pallina.

```
1 REM SUONO3
5 REM CAMBIAMENTO DINAMICO SENZA VOCE 3
100 REM  123456789012345678901234
110 DATA"                        "
120 DATA"                        "
130 DATA"         ******         "
140 DATA"       **********       "
150 DATA"      ************      "
160 DATA"     **************     "
170 DATA"    ****************    "
180 DATA"    ****************    "
190 DATA"   ******************   "
200 DATA"   ******************   "
210 DATA"   ******************   "
220 DATA"   ******************   "
230 DATA"   ******************   "
240 DATA"    ****************    "
250 DATA"    ****************    "
260 DATA"     **************     "
270 DATA"      ************      "
280 DATA"       **********       "
290 DATA"         ******         "
```

```
300 DATA"                                    "
310 DATA"                                    "
320 FORI=1TO21:READA$:PRINTA$
330 FORJ=0TO2:B$=MID$(A$,J*8+1,8)
340 BY=0:FORK=1TO8
350 IFMID$(B$,K,1)="*"THENBY=BY+2↑(8-K)
360 NEXT:POKE896+I*3+J,BY:NEXT:NEXT
370 POKE2040,14:POKE53269,1
380 POKE53248,10:PRINTCHR$(147)
390 V1=54272:V2=54279
400 POKEV1,0:POKEV1+1,0:POKEV2,0:POKEV2+1,10
410 POKEV1+5,0:POKEV1+6,240
415 POKEV2+5,8:POKEV2+6,0:POKEV2+3,8
420 POKEV1+4,33
430 POKEV1+24,15
440 G=2000
450 T=T+.03
460 YN=G*T*T+V0*T+S0:VN=(YN-Y)/.03:Y=YN
470 IFV<0ANDVN>0ANDY>231THENPOKEV1+1,0:END
480 V=VN
490 IFY>230THENRB=-1:Y=230
500 POKEV1+1,200-Y/3
510 POKE53249,Y:POKE53248,PEEK(53248)+2
515 POKEV1+24,17-PEEK(53248)/16
520 IFRBTHENV=-SQR(.7*V*V):V0=V:S0=230:T=0
530 IFRBTHENRB=0:POKEV2+4,64:POKEV2+4,65
540 GOTO450
```

COMMENTO A SUONO3.
Linee 0-380: crea lo sprite per la pallina.
Linee 390-430: inizializza le due voci che verranno usate.
Linea 440: inizializza la costante G, che rappresenta la gravità per la simulazione della pallina.
Linea 450: inizia un ciclo il cui indice è la variabile T; essa verrà inizializzata nuovamente quando sarà verificata la condizione di linea 490, cioè ogni volta che la pallina toccherà il pavimento.
Linea 460: calcola, in funzione della gravità, della velocità iniziale e del valore iniziale della Y, il valore della Y e della velocità attuale.
Linea 470: calcola la condizione in cui la pallina è ferma ed eventualmente esce.
Linea 490: se è verificata una condizione di rimbalzo, allora la variabile booleana RB viene posta a VERO.

Linea 500: la frequenza della voce 1 è proporzionale alla quota della pallina.
Linea 510: aggiorna la posizione dello sprite.
Linea 520: il volume è proporzionale alla posizione orizzontale della pallina.
Linee 520-530: se è verificata la condizione di rimbalzo, allora la velocità cambia di segno e viene ridotta in modulo (la pallina non è perfettamente elastica), vengono ripristinate le condizioni iniziali e la voce 2 ha il compito di produrre il rumore della pallina che colpisce il pavimento.
Linea 540: chiude il ciclo.

Seguono i programmi SUONO5, SUONO6, SUONO7, SUONO8, SUONO9 e SUONO10, che producono effetti sonori speciali.

```
1 REM SUONO5
5 REM MANIA DEL CALCOLATORE
10 SI=54272
20 POKESI+6,240:POKESI+5,0
25 POKESI+3,8:POKESI+4,65
30 POKESI+24,15
40 FORL=1TO100
50 POKESI+1,RND(1)*256
60 FORM=1TO50:NEXT
70 NEXT
80 POKESI+1,0:POKESI+24,0
```

```
1 REM SUONO6
1000 REM SIRENA
1010 VV=54272:POKEVV+6,0:POKEVV+5,237
1015 POKEVV+24,15
1020 POKEVV+2,200:POKEVV+3,0
1025 AA=55:BB=42
1030 POKEVV+1,60:POKEVV+4,65
1040 FOR NN=1TO12
1050 POKEVV+1,AA
1060 FORMM=1TO300:NEXTMM
1070 POKEVV+1,BB
1080 FORMM=1TO300:NEXTMM
1085 IF NN=6THEN AA=53:BB=40
1090 NEXT NN
1100 POKEVV+4,0
1120 POKEVV+24,0
```

```
1 REM SUONO7
5 REM ALLARME ROSSO
10 SI=54272
20 POKESI+6,240:POKESI+5,0
25 POKESI+3,8:POKESI+4,65
30 POKESI+24,15
40 FORL=1TO10
50 FORM=10TO50STEP2
60 POKESI+1,M
70 FORN=1TO10
80 NEXTN
90 NEXTM
100 POKESI+1,0
110 FORM=1TO100
120 NEXTM
130 NEXTL
140 POKESI+24,0
```

```
1 REM SUONO8
5 REM TELEFONO
10 SI=54272
20 POKESI+6,240:POKESI+5,0
25 POKESI+3,7:POKESI+4,65
30 POKESI+24,15
40 FORL=1TO5
50 FORM=1TO50
60 POKESI+1,47
70 FORN=1TO5
80 NEXTN
90 POKESI+1,0
100 NEXTM
110 FORM=1TO3000
120 NEXTM
130 NEXTL
140 POKESI+24,0
```

```
1 REM SUONO9
5 REM ONDE DEL MARE
10 SI=54272
20 POKESI+6,240:POKESI+5,0
25 POKESI+3,7:POKESI+4,129
30 POKESI+1,40
40 FORL=1TO10
50 D=INT(RND(1)*5)*50+50
60 FORM=3TO15
70 POKESI+24,M
80 FORN=1TOD
90 NEXTN
100 NEXTM
110 FORM=15TO3STEP-1
120 POKESI+24,M
130 FORN=1TOD
140 NEXTN
150 NEXTM
160 NEXTL
170 POKESI+1,0
180 POKESI+24,0
```

```
1 REM SUONO10
5 REM CINGUETTIO
10 SI=54272
20 POKESI+6,240:POKESI+5,0
25 POKESI+3,8:POKESI+4,65
30 POKESI+24,15
40 FORL=1TO20
50 FORM=210TO180+INT(RND(1)*20)STEP-2
60 POKESI+1,M
70 NEXTM
80 FORM=0TOINT(RND(1)*80)+80
90 NEXTM
100 NEXTL
110 POKESI+1,0
120 POKESI+24,0
```

APPENDICE A

I REGISTRI DEL VIC II

Riportiamo l'elenco dei registri del VIC II, indicando per ogni registro il numero, l'indirizzo decimale e esadecimale, e la funzione.

```
R E G I S T R I   D E L   V I C   I I

REG.INDIRIZZO      FUNZIONE

0    53248(D000H)  POS. X SPRITE NUM.0
1    53249(D001H)  POS. Y SPRITE NUM.0
2    53250(D002H)  POS. X SPRITE NUM.1
3    53251(D003H)  POS. Y SPRITE NUM.1
4    53252(D004H)  POS. X SPRITE NUM.2
5    53253(D005H)  POS. Y SPRITE NUM.2
6    53254(D006H)  POS. X SPRITE NUM.3
7    53255(D007H)  POS. Y SPRITE NUM.3
8    53256(D008H)  POS. X SPRITE NUM.4
9    53257(D009H)  POS. Y SPRITE NUM.4
10   53258(D00AH)  POS. X SPRITE NUM.5
11   53259(D00BH)  POS. Y SPRITE NUM.5
12   53260(D00CH)  POS. X SPRITE NUM.6
13   53261(D00DH)  POS. Y SPRITE NUM.6
14   53262(D00EH)  POS. X SPRITE NUM.7
15   53263(D00FH)  POS. Y SPRITE NUM.7
16   53264(D010H)
               BIT 7: MSB X SPRITE NUM.7
               BIT 6: MSB X SPRITE NUM.6
               BIT 5: MSB X SPRITE NUM.5
               BIT 4: MSB X SPRITE NUM.4
               BIT 3: MSB X SPRITE NUM.3
               BIT 2: MSB X SPRITE NUM.2
               BIT 1: MSB X SPRITE NUM.1
               BIT 0: MSB X SPRITE NUM.0
17   53265(D011H) REGISTRO DI CONTROLLO
               BIT 7: MSB REGISTRO DI LINEA
               BIT 6: 1 MODO SFONDO PROGRAMMABILE
```

```
REG.INDIRIZZO      FUNZIONE

          BIT 5: 1 MODO PAGINA GRAFICA
          BIT 4: 0 ANNULLAMENTO SCHERMO
          BIT 3: 0 SCHERMO A 24 RIGHE
          BIT 2-0: POS. Y DELLE SCRITTE
                   (SCORRIMENTO FINE)
18  53266(D012H) REGISTRO DI LINEA
19  53267(D013H) POS. X LIGHT-PEN
20  53268(D014H) POS. Y LIGHT-PEN
21  53269(D015H) ABILITAZIONE SPRITE
          BIT 7: 1 SPRITE NUM.7
          BIT 6: 1 SPRITE NUM.6
          BIT 5: 1 SPRITE NUM.5
          BIT 4: 1 SPRITE NUM.4
          BIT 3: 1 SPRITE NUM.3
          BIT 2: 1 SPRITE NUM.2
          BIT 1: 1 SPRITE NUM.1
          BIT 0: 1 SPRITE NUM.0
22  53270(D016H) REGISTRO DI CONTROLLO
          BIT 7-5: NON USATI
          BIT 4: 1 MODO MULTICOLORE
          BIT 3: 0 SCHERMO A 38 COLONNE
          BIT 2-0: POS. X DELLE SCRITTE
                   (SCORRIMENTO FINE)
23  53271(D017H) ESPANSIONE VERTICALE SPRITE
          BIT 7: 1 SPRITE NUM.7
          BIT 6: 1 SPRITE NUM.6
          BIT 5: 1 SPRITE NUM.5
          BIT 4: 1 SPRITE NUM.4
          BIT 3: 1 SPRITE NUM.3
          BIT 2: 1 SPRITE NUM.2
          BIT 1: 1 SPRITE NUM.1
          BIT 0: 1 SPRITE NUM.0
24  53272(D018H) REGISTRO CONTROLLO MEMORIA
          BIT 7-4: IND. BASE MAPPA VIDEO
          BIT 3-1: IND. BASE MAPPA CARATTERI
          BIT 0: NON USATO
25  53273(D019H) REGISTRO DI STATO INTERRUPT
          BIT 7: 1 INTERRUPT LANCIATO
                   DAL VIC II
```

```
REG.INDIRIZZO     FUNZIONE
          BIT 6-4: NON USATI
          BIT 3: 1 INTERRUPT LANCIATO
                   DALLA LIGHT-PEN
          BIT 2: 1 INTERRUPT COLLISIONE
                   TRA SPRITE
          BIT 1: 1 INTERRUPT COLLISIONE
                   SPRITE-SFONDO
          BIT 0: 1 INTERRUPT REGISTRO DI LINEA
26  53274(D01AH) REGISTRO ABILITAZIONE
                 INTERRUPT: 1=ABILITATO
          (UTILIZZO DEI BIT ANALOGO AL
           REGISTRO PRECEDENTE)
27  53275(D01BH) PRIORITA' SPRITE-SFONDO
          (1-SFONDO, 0-SPRITE)
          BIT 7: SPRITE NUM.7
          BIT 6: SPRITE NUM.6
          BIT 5: SPRITE NUM.5
          BIT 4: SPRITE NUM.4
          BIT 3: SPRITE NUM.3
          BIT 2: SPRITE NUM.2
          BIT 1: SPRITE NUM.1
          BIT 0: SPRITE NUM.0
28  53276(D01CH) SPRITE MULTICOLORE
          BIT 7: 1 SPRITE NUM.7
          BIT 6: 1 SPRITE NUM.6
          BIT 5: 1 SPRITE NUM.5
          BIT 4: 1 SPRITE NUM.4
          BIT 3: 1 SPRITE NUM.3
          BIT 2: 1 SPRITE NUM.2
          BIT 1: 1 SPRITE NUM.1
          BIT 0: 1 SPRITE NUM.0
29  53277(D01DH) ESPANSIONE ORIZZONTALE SPRITE
          BIT 7: 1 SPRITE NUM.7
          BIT 6: 1 SPRITE NUM.6
          BIT 5: 1 SPRITE NUM.5
          BIT 4: 1 SPRITE NUM.4
          BIT 3: 1 SPRITE NUM.3
          BIT 2: 1 SPRITE NUM.2
          BIT 1: 1 SPRITE NUM.1
          BIT 0: 1 SPRITE NUM.0
```

```
REG. INDIRIZZO     FUNZIONE

 30  53278(D01EH) COLLISIONE SPRITE-SPRITE
           BIT 7: 1 SPRITE NUM.7
           BIT 6: 1 SPRITE NUM.6
           BIT 5: 1 SPRITE NUM.5
           BIT 4: 1 SPRITE NUM.4
           BIT 3: 1 SPRITE NUM.3
           BIT 2: 1 SPRITE NUM.2
           BIT 1: 1 SPRITE NUM.1
           BIT 0: 1 SPRITE NUM.0
 31  53279(D01FH) COLLISIONE SPRITE-SFONDO
           BIT 7: 1 SPRITE NUM.7
           BIT 6: 1 SPRITE NUM.6
           BIT 5: 1 SPRITE NUM.5
           BIT 4: 1 SPRITE NUM.4
           BIT 3: 1 SPRITE NUM.3
           BIT 2: 1 SPRITE NUM.2
           BIT 1: 1 SPRITE NUM.1
           BIT 0: 1 SPRITE NUM.0
 32  53280(D020H) COLORE DEL BORDO
 33  53281(D021H) COLORE DELLO SFONDO
                  (COLORE DI SFONDO 1)
 34  53282(D022H) COLORE DI SFONDO 2
 35  53283(D023H) COLORE DI SFONDO 3
 36  53284(D024H) COLORE DI SFONDO 4
 37  53285(D025H) COLORE 0 SPRITE MULTICOLORE
 38  53286(D026H) COLORE 1 SPRITE MULTICOLORE
 39  53287(D027H) COLORE SPRITE NUM.0
 40  53288(D028H) COLORE SPRITE NUM.1
 41  53289(D029H) COLORE SPRITE NUM.2
 42  53290(D02AH) COLORE SPRITE NUM.3
 43  53291(D02BH) COLORE SPRITE NUM.4
 44  53292(D02CH) COLORE SPRITE NUM.5
 45  53293(D02DH) COLORE SPRITE NUM.6
 46  53294(D02EH) COLORE SPRITE NUM.7
```

NOTA: nella tabella MSB (Most Significative Bit) sta per BIT PIU' SIGNIFICATIVI.

I REGISTRI DEL SID

Riportiamo l'elenco dei registri del SID, indicando per ogni registro il numero, l'indirizzo decimale e esadecimale, e la funzione.

```
R E G I S T R I   D E L   S I D

REG. INDIRIZZO        FUNZIONE

0    54272(D400H)    LO FREQUENZA VOCE 1
1    54273(D401H)    HI FREQUENZA VOCE 1
2    54274(D402H)    LO LARGHEZZA ONDA
                     IMPULSIVA VOCE 1
3    54275(D403H)
           BIT 7-4:  NON USATI
           BIT 3-0:  NIBBLE PIU' SIGN. LARGHEZZA
                     ONDA IMPULSIVA VOCE 1
4    54276(D404H)    REGISTRO CONTROLLO VOCE 1
           BIT 7:  1 SELEZIONA FORMA ONDA
                     CASUALE (RUMORE)
           BIT 6:  1 SELEZIONA FORMA ONDA
                     IMPULSIVA
           BIT 5:  1 SELEZIONA FORMA ONDA
                     A DENTE DI SEGA
           BIT 4:  1 SELEZIONA FORMA ONDA
                     TRIANGOLARE
           BIT 3:  1 OSCILLATORE 1 DISABILITATO
           BIT 2:  1 OSCILL. 1 MODULATO CON
                     USCITA OSCILLATORE 3
           BIT 1:  1 OSCILL. 1 SINCRONIZZATO
                     CON FREQUENZA OSCILL. 3
           BIT 0:  1 PARTENZA ATT./DEC./SUS.
                   0 PARTENZA RELEASE
```

```
REG.INDIRIZZO      FUNZIONE

 5   54277(D405H)
           BIT 7-4: SELEZIONA ATTACK GENERATORE 1
           BIT 3-0: SELEZIONA DECAY GENERATORE 1
 6   54278(D406H)
           BIT 7-4: SELEZIONA SUSTAIN GENERATORE 1
           BIT 3-0: SELEZIONE RELEASE GENERATORE 1
 7   54279(D407H)   LO FREQUENZA VOCE 2
 8   54280(D408H)   HI FREQUENZA VOCE 2
 9   54281(D409H)   LO LARGHEZZA ONDA
                    IMPULSIVA VOCE 2
 10  54282(D40AH)
           BIT 7-4: NON USATI
           BIT 3-0: NIBBLE PIU' SIGN. LARGHEZZA
                    ONDA IMPULSIVA VOCE 2
 11  54283(D40BH)   REGISTRO CONTROLLO VOCE 2
           BIT 7: 1 SELEZIONA FORMA ONDA
                    CASUALE (RUMORE)
           BIT 6: 1 SELEZIONA FORMA ONDA
                    IMPULSIVA
           BIT 5: 1 SELEZIONA FORMA ONDA
                    A DENTE DI SEGA
           BIT 4: 1 SELEZIONA FORMA ONDA
                    TRIANGOLARE
           BIT 3: 1 OSCILLATORE 2 DISABILITATO
           BIT 2: 1 OSCILL. 2 MODULATO CON
                    USCITA OSCILLATORE 1
           BIT 1: 1 OSCILL. 2 SINCRONIZZATO
                    CON FREQUENZA OSCILL. 1
           BIT 0: 1 PARTENZA ATT./DEC./SUS
                  0 PARTENZA RELEASE
 12  54284(D40CH)
           BIT 7-4: SELEZIONA ATTACK GENERATORE 2
           BIT 3-0: SELEZIONA DECAY GENERATORE 2
 13  54285(D40DH)
           BIT 7-4: SELEZIONA SUSTAIN GENERATORE 2
           BIT 3-0: SELEZIONA RELEASE GENERATORE 2
 14  54286(D40EH)   LO FREQUENZA VOCE 3
 15  54287(D40FH)   HI FREQUENZA VOCE 3
 16  54288(D410H)   LO LARGHEZZA ONDA IMPULSIVA
                    VOCE 3
```

```
REG.INDIRIZZO      FUNZIONE
17  54289(D411H)
          BIT 7-4: NON USATI
          BIT 3-0: NIBBLE PIU' SIGN. LARGHEZZA
                   ONDA IMPULSIVA VOCE 3
18  54290(D412H)   REGISTRO CONTROLLO VOCE 3
          BIT 7: 1 SELEZIONA FORMA ONDA
                   CASUALE (RUMORE)
          BIT 6: 1 SELEZIONA FORMA ONDA
                   IMPULSIVA
          BIT 5: 1 SELEZIONA FORMA ONDA
                   A DENTE DI SEGA
          BIT 4: 1 SELEZIONA FORMA ONDA
                   TRIANGOLARE
          BIT 3: 1 OSCILLATORE 3 DISABILITATO
          BIT 2: 1 OSCILL. 3 MODULATO CON
                   USCITA OSCILLATORE 2
          BIT 1: 1 OSCILL. 3 SINCRONIZZATO
                   CON FREQUENZA OSCILL. 2
          BIT 0: 1 PARTENZA ATT./DEC./SUS
                 0 PARTENZA RELEASE
19  54291(D413H)
          BIT 7-4: SELEZIONA ATTACK GENERATORE 3
          BIT 3-0: SELEZIONA DECAY GENERATORE 3
20  54292(D414H)
          BIT 7-4: SELEZIONA SUSTAIN GENERATORE 3
          BIT 3-0: SELEZIONA RELEASE GENERATORE 3
21  54293(D415H)
          BIT 7-3: NON USATI
          BIT 2-1: LSB FREQUENZA DI TAGLIO
                   DEL FILTRO
22  54294(D416H)   HI FREQUENZA DI TAGLIO
                   DEL FILTRO
23  54295(D417H)
          BIT 7-4: RISONANZA FILTRO
          BIT 5: 1 FILTRA INPUT ESTERNO
```

```
REG. INDIRIZZO       FUNZIONE
24   54296(D418H)
               BIT 7:  0 USCITA VOCE 3 DISABILITATA
               BIT 6:  1 FILTRO PASSA ALTO
               BIT 5:  1 FILTRO PASSA BANDA
               BIT 4:  1 FILTRO PASSA BASSO
               BIT 3-0: VOLUME
25   54297(D419H)    CONVERTITORE ANALOGICO
                     -DIGITALE (PADDLE 1)
26   54298(D41AH)    CONVERTITORE ANALOGICO
                     -DIGITALE (PADDLE 2)
27   54299(D41BH)    USCITA GENERATORE 3
28   54300(D41CH)    USCITA INVILUPPO GENERATORE 3
```

NOTA: nella tabella valgono le seguenti abbreviazioni:

LO byte meno significativo di un numero formato da 2 byte;
HI byte più significativo di un numero formato da 2 byte;
MSB bit più significativi di un numero espresso in binario;
LSB bit meno significativi di un numero espresso in binario;
NIBBLE mezzo byte, cioè 4 bit.

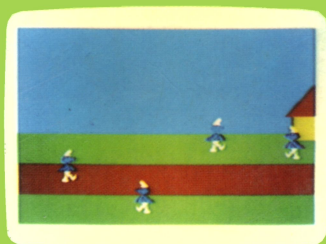
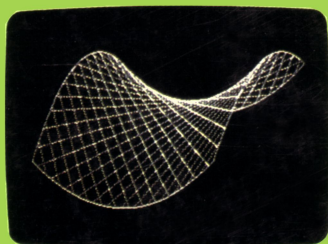
VALORI DELLE NOTE

Riportiamo i valori da porre nel byte più significativo (HI) e nel byte meno significativo (LO) del registro della frequenza per ottenere una nota musicale per 8 ottave.

T A B E L L A D E L L E N O T E

NOTA	FREQ.	VAL.	HI	LOW
DO-0	16.4	278	1	22
DO#-0	17.3	295	1	39
RE-0	18.4	313	1	57
RE#-0	19.4	331	1	75
MI-0	20.6	351	1	95
FA-0	21.8	372	1	116
FA#-0	23.1	394	1	138
SOL-0	24.5	417	1	161
SOL#-0	26	442	1	186
LA-0	27.5	468	1	212
LA#-0	29.1	496	1	240
SI-0	30.9	526	2	14
DO-1	32.7	557	2	45
DO#-1	34.6	590	2	78
RE-1	36.7	625	2	113
RE#-1	38.9	662	2	150
MI-1	41.2	702	2	190
FA-1	43.7	743	2	231
FA#-1	46.2	788	3	20
SOL-1	49	834	3	66
SOL#-1	51.9	884	3	116
LA-1	55	937	3	169
LA#-1	58.3	992	3	224

NOTA	FREQ.	VAL.	HI	LOW
SI-1	61.7	1051	4	27
DO-2	65.4	1114	4	90
DO#-2	69.3	1180	4	156
RE-2	73.4	1250	4	226
RE#-2	77.8	1324	5	44
MI-2	82.4	1403	5	123
FA-2	87.3	1487	5	207
FA#-2	92.5	1575	6	39
SOL-2	98	1669	6	133
SOL#-2	103.8	1768	6	232
LA-2	110	1873	7	81
LA#-2	116.5	1985	7	193
SI-2	123.5	2103	8	55
DO-3	130.8	2228	8	180
DO#-3	138.6	2360	9	56
RE-3	146.8	2500	9	196
RE#-3	155.6	2649	10	89
MI-3	164.8	2807	10	247
FA-3	174.6	2973	11	157
FA#-3	185	3150	12	78
SOL-3	196	3338	13	10
SOL#-3	207.7	3536	13	208
LA-3	220	3746	14	162
LA#-3	233.1	3969	15	129
SI-3	246.9	4205	16	109
DO-4	261.6	4455	17	103
DO#-4	277.2	4720	18	112
RE-4	293.7	5001	19	137
RE#-4	311.1	5298	20	178
MI-4	329.6	5613	21	237
FA-4	349.2	5947	23	59
FA#-4	370	6300	24	156
SOL-4	392	6675	26	19
SOL#-4	415.3	7072	27	160
LA-4	440	7493	29	69
LA#-4	466.2	7938	31	2
SI-4	493.9	8410	32	218
DO-5	523.3	8910	34	206
DO#-5	554.4	9440	36	224
RE-5	587.3	10001	39	17
RE#-5	622.3	10596	41	100

NOTA	FREQ.	VAL.	HI	LOW
MI-5	659.3	11226	43	218
FA-5	698.5	11894	46	118
FA#-5	740	12601	49	57
SOL-5	784	13350	52	38
SOL#-5	830.6	14144	55	64
LA-5	880	14985	58	137
LA#-5	932.3	15876	62	4
SI-5	987.8	16820	65	180
DO-6	1046.5	17820	69	156
DO#-6	1108.7	18880	73	192
RE-6	1174.7	20003	78	35
RE#-6	1244.5	21192	82	200
MI-6	1318.5	22452	87	180
FA-6	1396.9	23787	92	235
FA#-6	1480	25202	98	114
SOL-6	1568	26700	104	76
SOL#-6	1661.2	28288	110	128
LA-6	1760	29970	117	18
LA#-6	1864.7	31752	124	8
SI-6	1975.5	33640	131	104
DO-7	2093	35641	139	57
DO#-7	2217.5	37760	147	128
RE-7	2349.3	40005	156	69
RE#-7	2489	42384	165	144
MI-7	2637	44904	175	104
FA-7	2793.8	47574	185	214
FA#-7	2960	50403	196	227
SOL-7	3136	53400	208	152
SOL#-7	3322.4	56576	221	0
LA-7	3520	59940	234	36
LA#-7	3729.3	63504	248	16

Questo libro tratta in maniera approfondita la grafica e il suono. Ogni argomento viene spiegato e accompagnato da numerosi esempi commentati. Nel libro sono listati moltissimi programmi; essi sono registrati sulla cassetta allegata. Inoltre la cassetta contiene anche altri programmi, ai quali si fa riferimento nel testo, ma senza listarli. Sono presenti anche routine in linguaggio macchina per la grafica, che possono essere riutilizzate anche in altri programmi.
Nel Capitolo 1 si tratta del video, della tastiera e della grafica in modo carattere. Il Capitolo 2 è dedicato alla grafica e ne approfondisce tutti gli aspetti. Il Capitolo 3 tratta degli sprite e gradualmente porta a costruire giochi di animazione in BASIC. Nel Capitolo 4 viene trattato l'argomento del suono. Completano il libro tre appendici: i registri del VIC II e del SID e le frequenze delle note musicali.

L. 34.000 Cod. 409B ISBN88-7056-170-4

137

COMMODORE 64 la grafica e il suono

Rita Bonelli
Luciano Pazzucconi
Fabio Racchi
Giovanni Valerio

GRUPPO EDITORIALE JACKSON